An Analysis of the Applicability of RSVP

Ursula Schwantag

Diploma Thesis at the Institute of Telematics

Prof. Dr. Dr. h. c. Gerhard Krüger

Fakultät für Informatik

Universität Karlsruhe (TH)

Advisers:

Prof. Dr. Dr. h. c. G. Krüger

Dr.-Ing. Claudia Schmidt	David Meyer, M.S.
Institut für Telematik	Advanced Network Technology Center
Universität Karlsruhe	University of Oregon

July 15, 1997

Erklärung

Ich erkläre hiermit, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 15. Juli 1997

Contents

1 Introduction **1**

2 Integrated Services **5**

2.1 Building blocks of real-time traffic 5

2.1.1 Playback applications and quality of service parameter 5

2.1.2 Multicast 7

2.1.3 RTP - Real-Time Protocol 8

2.2 Types of service in Integrated Services 10

2.2.1 Traffic and application classification 10

2.2.2 Service commitments 12

2.2.3 Implementation of traffic control 14

2.3 RSVP protocol overview 15

2.3.1 RSVP design features 17

2.3.2 Applicability as intended by the RSVP working group 20

3 Quality of service in the current Internet model **21**

3.1 Measuring quality of service parameters 21

3.2 Causes of jitter 24

3.2.1 Bursty traffic and queues 24

3.2.2 Experiments: Host-induced jitter 26

3.2.3 Experiments: Router-induced jitter 31

3.2.4 Implications of jitter not induced by queuing 37

3.3 Causes of packet loss 39

3.3.1 Effects of packet loss in audio and video applications 39

3.4 MBone traffic characterization 41

3.5 Summary of Experiments 45

4 Evaluation of Integrated Services and RSVP 47

4.1 General discussion of RSVP 47

4.2 Experiments: Reservations on small interfaces 49

4.2.1 Experiment setup 49

4.2.2 The effect of reservations on packet loss 51

4.2.3 The effect of reservations on jitter 52

4.2.4 Effects during reservation setup 54

4.3 Lower layers and reservation enforcement 56

4.3.1 Link layer QoS in distribution networks and backbones 56

4.3.2 Integrated Services on LANs 57

4.4 Fundamental problems with guarantees 58

4.4.1 The service contract of controlled load service 59

4.4.2 The service contract of guaranteed service 59

4.5 Scaling issues of Integrated Services/RSVP 61

5 Approaches to solve the scaling problems of Integrated Services/RSVP 65

5.1 Aggregation 65

5.1.1 "Classy" aggregation 66

5.1.2 "Classy" aggregation with full reservation state 68

5.1.3 Hierarchical RSVP 70

5.2 Switching 71

5.2.1 IP switching 71

5.2.2 Tag switching 76

5.2.3 NetFlow switching 79

5.3 Non-RSVP clouds with static QoS 79

5.4 Summary of approaches 81

6 Alternatives to RSVP and Integrated Services 83

6.1 Adding bandwidth and over-provisioning 85

6.2 Volume-based billing 86

6.3 Class of service approaches with precedence classes 87

6.4 Static resource allocation in advance 90

6.5 ST-II 91

7 Conclusions **93**

A List of Acronyms **101**

Chapter 1

Introduction

On today's Internet, "real-time" applications are playing an important role. Computer networks used to carry computer data only. Today, they are also used for internet telephony, multimedia conferencing, transmission of lectures and meetings, distributed simulations, network games and other real-time applications. Future application areas with real-time requirements might include telemedicine, distributed workgroups, distance learning and telecommuting.

At first glance, datagram networks do not seem suitable for real-time traffic. Packets are routed independently across shared networks, so transit times vary significantly. Variations in transit delays are called *jitter*. A class of real-time applications called *playback applications* aims to solve the jitter problem by buffering. *Adaptive playback applications* adapt to changing delays and thus work well on moderately loaded datagram networks. They can deal with jitter caused by short-lived bursts, and they can tolerate occasional lost packets during brief periods of congestion.

However, parts of the Internet are often heavily loaded. The price tag attached to sharing bandwidth is congestion, leading to jitter and packet loss. We have seen that at certain times of the day, some MBone audio multicasts ([Kum95]) are unintelligible because of more than 30% packet loss. While real-time traffic contributes heavily to congestion because of large bandwidth requirements, it also suffers more from congestion than non-real-time traffic. The only effect of congestion on non-real-time traffic is that a transfer takes longer to complete. In contrast, real-time data becomes obsolete if it doesn't arrive in time. As a consequence, real-time applications deliver poor quality during periods of congestion.

The underlying problem is that different classes of applications require different services. For example, a file transfer application requires that some quantity of data is transferred in an acceptable amount of time, while internet telephony requires that most packets get to the receiver in less than 0.3 seconds. If enough bandwidth is available, best-effort service fulfills all of these requirements. When resources are scarce, however, real-time traffic should be treated differently. The *Integrated Services* working group in the IETF (Internet Engineering Task Force) developed an enhanced Internet service model that includes best-effort service and real-time service ([BCS94]). Together with the reservation setup protocol RSVP ([ZDE$^+$93]), this architecture is a comprehensive approach to provide applications with the type of service they need in the quality they choose. Integrated Services are going to be used as follows: If best-effort service doesn't lead to satisfying results, a user can decide to make a reservation. If there is enough reservable bandwidth available on the path, the reservation is accepted. Otherwise it is rejected, corresponding to a busy signal, and the user keeps getting only best-effort service. Depending on the type of service chosen, a real-time application will get a certain guaranteed bandwidth with a bound on delay or a service that is as good as if the network was lightly loaded.

To honor reservations, routers on the path need to distinguish between best-effort flows and reserved flows. A *flow* is a stream of related packets from one source to a unicast or multicast destination. Routers use sophisticated scheduling techniques to provide service according to the reservations and to other policies. Reservations don't come for free, however. Reservation setup, management and enforcement consume bandwidth and processing power and can cause scalability problems.

This thesis examines the potential of Integrated Services to improve the quality of service for real-time applications. A first task was to analyze the causes of insufficient quality of service under the current service model. This task included developing a measurement method and using and creating tools to measure quality of service parameters. Both theoretical analysis and experiments were to be used to identify conceptual difficulties in deploying Integrated Services/RSVP. Various properties of existing networks are examined that limit the applicability of Integrated Services and RSVP as designed by the IETF working group. As a second part of this thesis, possible solutions for some of the open issues of RSVP were to be researched and developed. Especially the scalability problem of RSVP was to be examined. Finally, alternatives to Integrated Services and RSVP were to be found that provide a simpler, but effective solution to the QoS problem for real-time applications.

This document is organized as follows. Chapter 2 presents a new Internet service model, the Integrated Services model. Section 2.1 explains the mechanisms needed to efficiently use a datagram network for real-time traffic. Section 2.2 presents the different services in the Integrated Services model. A brief overview of the protocol architecture of RSVP is given in Section 2.3. Chapter 3 describes our experiments, measurement methods and observations about quality of service in the current service model. Chapter 4 discusses characteristics and problems of RSVP. The general advantages and disadvantages of resource reservation and RSVP are discussed in Section 4.1. We conducted experiments on the effect of reservations on packet loss and jitter in a controlled test environment. The results are presented in Section 4.2. In Section 4.3, we discuss the difficulties of controlling QoS on the link layer. Concerns about the usefulness of quantitative end-to-end guarantees are raised in Section 4.4. Section 4.5 describes the scalability problems that stand in the way of an efficient implementation of Integrated Services on high-performance routers. Chapter 5 presents aggregation and switching as two approaches to solve these scalability problems. We develop a new form of aggregation and examine the applicability of two commercial switching approaches that were designed with goals other than RSVP in mind. Chapter 6 presents alternative strategies to improve the quality of service for real-time applications. A promising approach is differentiated service based on precedence levels, described in Section 6.3. Chapter 7 summarizes our results and concludes the document.

Chapter 2

Integrated Services

This introductory chapter first explains the mechanisms needed to use a datagram network for real-time multimedia sessions. Subsequent sections present the concepts of Integrated Services and give an overview of the reservation protocol RSVP.

2.1 Building blocks of real-time traffic

2.1.1 Playback applications and quality of service parameter

The Integrated Services working group believes that most present and future real-time applications are *playback applications* . If they are not, they can probably be treated as such. The most common playback applications are audio and video tools. Well-known examples are the audio tool vat ([JM]) and the video tool vic ([MJ95]). In a playback application, a source samples a signal, packetizes the data and transmits it over the network ([CSZ92]). The receiver then depacketizes the data and plays it back. In a datagram network, transit times vary. If the receiver played the contents of the packets immediately upon arrival, the signal would be distorted. For that reason, the receiver buffers all incoming packets. At a designated playback point, the receiver attempts to faithfully replay the signal. Data that arrives at any time before the playback point is used to reconstruct the signal. Data that doesn't arrive in time leads to gaps in the replayed signal. A packet that arrives after its playback point is useless and has to be discarded. If the maximum delay was known in advance, the application could set its playback point to exactly the maximum delay so no packets would be late or lost. However, a trade-off between interactivity and playback quality has to be considered in determining the playback point. For interactive communication like in an audio conference, the total delay must be limited. A delay of only 250 to 300 ms is noticeable in a telephone conversation ([Sta88]) and is slightly irritating for the participants. The absolute delay doesn't matter if there is no interaction between sender and receiver, for example in a TV transmission. Some audio tools support this distinction and allow the user to switch between interactive mode and lecture mode.

In "traditional" technologies for real-time services (telephone lines, cable TV, radio waves) there is no delay variation. The delay is the constant time it takes for a signal to traverse the medium and intermediate switches. It only depends on the static properties of the medium. This desirable property is achieved by not having to share the medium. Every resource (wire, frequency, time slot) is explicitly assigned to one sender or sender/receiver pair. By contrast, sharing is a basic principle of all packet-switching networks. The consequence of sharing is queuing, a mechanism to deal with packet bursts. Arriving packets are queued when packets temporarily come in faster than they can be forwarded. As queues grow and shrink, packets experience different delays and thus different transit times between sender and receiver. We speak of *delay variation* or *jitter*. If the demand for network resources such as router processing time and bandwidth exceeds the available capacity over some period of time, a network is said to be *congested*. Congestion leads to high jitter and packet loss. Packet loss can also be thought of as infinite delay variation.

Delay, jitter and bandwidth are the *quality of service* (QoS) parameters that need to be controlled to support QoS for playback applications. Enough bandwidth must be available on the path for the average transmit rate plus a little extra so bursts can be cleared quickly. If the bandwidth is not sufficient, packets are dropped as soon as the queuing capacity is exhausted.

Packet delays need to be limited so most packets arrive in time for their playback point. However, the absolute delay can only be partially controlled. The total delay is composed of a fixed part and a variable part. The fixed part is the transit time through the medium at the speed of light plus the processing time in the switches. Nothing can be done about the fixed transit time. The variable delay is the time a packet spends in service queues. Packet scheduling techniques can influence delay variations. Thus, a bound on jitter implicitly limits the packet delay.

Section 3.1 explains how quality of service parameters are measured.

2.1.2 Multicast

IP multicast allows loosely-coupled multimedia sessions with large numbers of receivers. IP multicast is a connectionless model, because connection-oriented approaches to multimedia conferencing have severe disadvantage ([Jac94]). Connection-oriented multimedia applications do not scale with the number of participants in a session. For n sending participants, there need to be $O(n^2)$ connections. Joining and leaving in-progress conferences is difficult, because every participant needs to know all other participants. When links go down and come up, applications have to handle a large amount of connection management. IP multicast avoids these problems, but sacrifices reliability.

A multicast session is identified by a multicast group address. Any host can send to a multicast group simply by specifying the multicast group address as the destination address. A sender does not need to know anything about the receivers, neither their number nor their location or IP addresses ([Dee91]). A participant joins a session by joining the respective multicast group. All traffic destined for that group is automatically sent to the participant. Thus, joining and leaving a conference is reduced to joining and leaving a multicast group, which is handled by lower layers. Since there are no connections, intermittent connectivity does not lead to any overhead on the part of the application.

Multicast packets are routed along trees that include all the receivers as leaves. Multicast routing procedures are responsible for constructing the trees. Routers in the tree know where to forward packets destined for the group. Multicast sources send out only one copy of a packet, while unicast sources send one copy for every receiver. There is only one copy of each multicast packet on each branch of the multicast distribution tree. Where branches split, a new copy is made. Thus, multicast minimizes the bandwidth consumed by multimedia applications. (The original work on multicast routing is [Dee91]. For an overview over multicast routing protocols see [Hui95] or [MS97].)

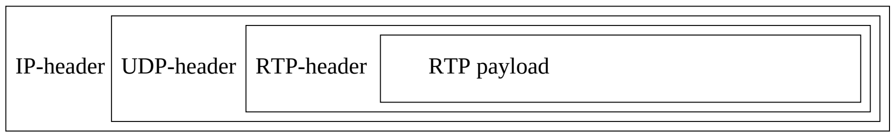

Figure 2.1: IP packet containing real-time data

Multicasting requires that multicast routing protocols are supported by all routers. The MBone (multicast backbone, see for example [Kum95]) is a subset of the Internet that supports multicasting. Isolated multicast-capable clouds are connected to the MBone through tunnels.

2.1.3 RTP - Real-Time Protocol

There are two transport layer protocols in the Internet protocol suite, TCP and UDP. TCP provides a reliable flow between two host. It is connection-oriented and thus can't be used for multicast. UDP provides a connectionless unreliable datagram service. To use UDP as a transport protocol for real-time traffic, some functionality has to be added. Functionality that is needed for many real-time applications is combined into RTP, the *real-time transport protocol*. RTP is standardized in RFC 1889 ([SCFJ96]). Applications typically run RTP on top of UDP as part of the transport layer protocol. Figure 2.1 shows the structure of an IP packet containing real-time data. In practice, RTP is usually implemented within the application. RTP is designed to be independent from the underlying transport protocol and can be used over unicast as well as multicast.

The services that RTP provides include timestamping, sequence numbering, payload identification and source identification. The most important information in the RTP header is the timing information. The sender timestamps each RTP packet with the point in time the first sample in the packet was encoded. The receiver then uses these timestamps to reconstruct the original timing before playing back the data. Timestamps contain relative timing information that represents timing relations between packets, not absolute points in time. Therefore, sender and receiver do not need to be synchronized.

Sequence numbers are necessary to handle out-of-order packets and to detect lost packets.

A payload type identifier tells the receiving application how to interpret the payload.

It identifies the format of the payload and specifies which encoding and compression schemes are used. Examples for payload formats are different versions of PCM, JPEG or H261. (See [Sch96] for a list of video and audio encoding standards and references.) The mapping between payload type codes and formats is specified in separate documents called profiles ([Sch96]).

In audio conferences, source identification allows the receiving application to indicate to the user who is talking at the moment. In video applications with several senders, all sources send to the same multicast address, so source identification is needed to assign incoming packets to the proper video image.

The format of the RTP header is shown below. Other fields include a version number, padding, extension and marker bits for special and experimental uses, and the number of contributing sources. This number is more than one if the payload of an RTP packet contains data from several sources. These sources are then listed as contributing sources. The synchronization source indicates where the data was combined, or it indicates the source of the data if there is only one.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X| CCount|M| payload type|       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The transport protocol RTP is augmented by RTCP, the *real-time control protocol.* Its primary function is to provide feedback on the quality of the data distribution. Applications may use this feedback to adapt to different network conditions, e.g. by using adaptive encoding. Feedback about transmission quality is also useful to locate problems and diagnose faults. Another function of RTCP is to keep track of participants when their internal identifiers change, and to distribute information about all participants in a session, for example their names and email addresses. Control packets are periodically transmitted to all participants in the session. Every control packet contains

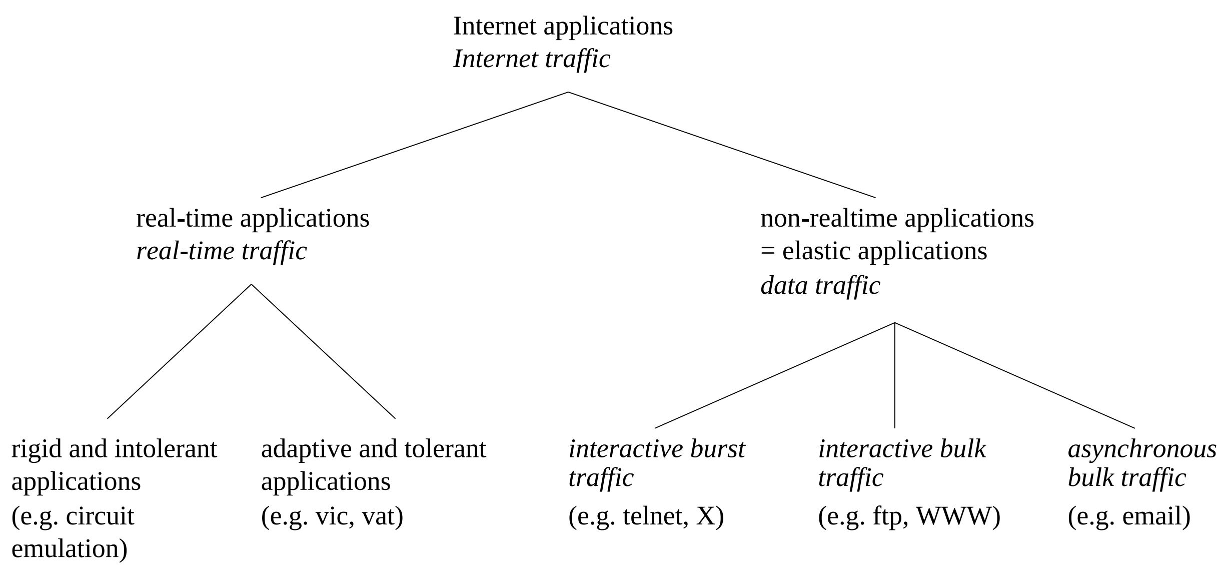

Figure 2.2: Traffic and application classification

reception statistics in the form of receiver reports. Receiver reports are issued by every receiver for every source. They report the number and fraction of lost packets and the interarrival jitter as well as to which packets these statistics refer. In this work, we use the same method to measure and calculate delay variations and jitter as specified for the RTCP reports.

2.2 Types of service in Integrated Services

2.2.1 Traffic and application classification

The two fundamentally different traffic types on datagram networks are *real-time traffic* and *non-realtime traffic*, for lack of a better name also called *data traffic*. [BCS94] distinguishes them as follows.

Data traffic originates from applications like telnet, ftp, WWW, email etc. These are *elastic applications*: they always wait for all data to arrive. Long delays degrade performance, but the final outcome of the data transfer is not affected by unfavorable network conditions. Elastic applications can be further broken up according to their delay requirements. *Interactive burst traffic* (telnet, X, NFS) needs short delay, *inter-*

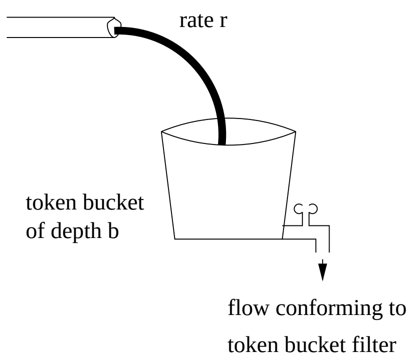

Figure 2.3: The token bucket model

active bulk traffic (FTP, WWW) requires medium delay, and *asynchronous bulk traffic* (email) works fine with high delay.

Data traffic is sporadic and unpredictable. Connections are usually short-lived and serve to transfer one or a few portions of data. Applications send out data as fast as their connection allows, then stop. Thus, data traffic is bursty.

Real-time traffic originates from *real-time applications*. Real-time applications are actually only quasi-real-time applications. All so-called real-time applications tolerate small delays. But in contrast to elastic applications, they are sensitive to higher delays and delay variations. The packet generation process of real-time applications often is regular and long lasting. Audio flows are especially predictable. They usually have a constant packet rate and fixed-size packets. For video, the coding algorithm as well as the nature of the image determines the shape of the flow. A *token bucket filter* can be used to characterize a real-time flow. A token bucket is determined by two parameters, a rate r and a depth b. In the model illustrated in Figure 2.3, the bucket is continuously filling with tokens at rate r. There can be at most b tokens in the bucket. Every time the source sends out a packet of size p, p tokens drain out of the bucket. A source that conforms to a token bucket filter (r, b) can only send when the bucket contains enough tokens. Thus, r is also the long-term maximum rate of the flow, and b is its burst size. For example, a well-behaving PCM audio source could be specified as having $r = 64\text{Kbit/s}$ and $b =$ size of one packet.

RFC 1633 ([BCS94]) and [CSZ92] further classify real-time applications into *rigid* and *adaptive applications* . Rigid applications have a fixed playback point, while adaptive applications adapt their playback point according to the current network conditions. Rigid applications use a delay bound advertised by the network to set their playback point. This bound is typically larger than the actual delay. Adaptive applications take advantage of the smaller actual delay, but they risk setting the playback point too early and losing packets. Thus, adaptive applications need to be tolerant of occasional packet losses. Examples of adaptive applications include video and audio tools. Rigid applications are usually intolerant. [BCS94] mentions circuit emulation as an example for an intolerant application.

2.2.2 Service commitments

Traditionally, IP service provides the same quality of service to every packet. Integrated Services expand this service model so that applications can choose an appropriate service ([BCS94]). The services that are specified so far are *best-effort service*, *guaranteed service* (specified in [SPG97]) and *controlled load service* (specified in [Wro97]). Other services, for example controlled delay, have been proposed but haven't gained acceptance.

Integrated Services also include *controlled link sharing* ([FJ95]). Often, multiple organizations share a link. They each pay a fixed share of the cost, thus they want to receive a guaranteed share of the link bandwidth during congestion. Moreover, bandwidth shouldn't go unused while any organization has data do send. Controlled link sharing is also desirable to assign a certain bandwidth to different protocol families. This avoids situations in which one protocol gets no bandwidth because of its back-off behavior during congestion. A third use is isolating different services, for example to ensure that network control traffic gets a guaranteed share of the bandwidth.

Best-effort service

Traffic from elastic applications gets best-effort service. The network promises nothing but tries to deliver packets as soon as possible. All real-time flows that do not have a reservation are also delivered with best-effort service.

Guaranteed quality of service

Guaranteed service is used by rigid intolerant applications. It provides a firm bound on delay and no packet loss for a flow that conforms to its token bucket specification. Guaranteed service does not attempt to minimize jitter. It merely controls the maximum queuing delay. An application can then set its playback point so that all packets arrive in time.

It has been proven that the queuing delay of a flow conforming to a token bucket (r, b) and being served by a line with bandwidth R is at most b/R as long as $r \leq R$ ([PG93]). Thus, if guaranteed service provides a guaranteed share of bandwidth $\geq r$, the queuing delay is bounded by the time the last packet of a burst of size b spends in the queue at one link. This result applies to the *fluid model*, which is the service that would be provided if there was a wire of bandwidth R between source and receiver. The bound on queuing delay needs to be adjusted by error terms that specify how the flow deviates from the fluid model. Then, the fixed latency of the path is added to get a bound on the absolute delay.

The application thus specifies its own traffic characteristics, receives the exported information on error terms and latency from the network and then decides whether the resulting delay bound is sufficient. If the application decides to accept the service, it sets its playback point to the maximum delay. The actual delay for the majority of packets will be much lower than the guaranteed delay. Therefore, packets will have to be buffered at the receiver.

Controlled load service

Controlled load service is the service designed for adaptive, tolerant applications. No quantitative guarantees are given, but the service under overload is about as good as best-effort service on a lightly loaded network.

A client provides the network with the token bucket specification of the traffic it will generate. The network ensures that enough resources will be available for that flow, as long as the flow conforms to the specification. The service given to non-conforming packets is not specified. The service for conforming packets is characterized by low delays and little packet loss. Queuing delays are not significantly larger than the time it takes to clear a maximum burst at the requested transmission rate. Occasional packet

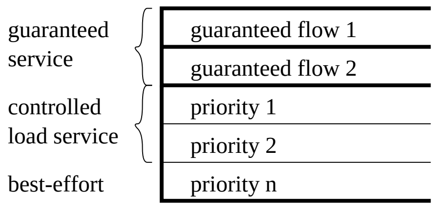

Figure 2.4: Hierarchical traffic control

losses may occur due to statistical effects, but the total loss rate must not greatly exceed the basic packet error rate of the transmission medium.

2.2.3 Implementation of traffic control

Every network element on a path has to support the distinction between different services. Otherwise, it is not guaranteed that the demanded end-to-end quality of service can be supported.

It is left to the router vendors to implement the specified services in network elements. Implementations may differ in the underlying mechanisms, processing speed and utilization of the bandwidth.

A possible implementation framework is given by the Integrated Services working group in [BCS94]. The scheduling algorithm *weighted fair queuing* (WFQ) is an important building block. WFQ is used to isolate flows from each other. Each WFQ flow has a separate queue and packets are scheduled so that each flow receives a constant fraction of the link bandwidth during congestion. In contrast to static time-slicing mechanisms, no bandwidth is wasted while there is demand. If a WFQ flow does not use its assigned bandwidth, other flows can use it. WFQ is computationally expensive because it dynamically determines the next queue to be served according to the bandwidth that each queue received previously. At first glance, weighted round robin seems to be a less complex alternative. With weighted round robin, every flow gets its turn to send a number of packets that is determined by the flow's weight. But weighted round robin is only fair in terms of packets sent by each flow, not in terms of bandwidth used by each flow. For that reason, weighted round robin cannot be used to isolate flows, and the more complex WFQ algorithm is used.

At the top level, each guaranteed service flow gets its own WFQ queue. Thus, guaranteed flows are strictly separated from each other and from the rest of the traffic. All other traffic is assigned to a pseudo WFQ flow. Within this flow, controlled load traffic and best-effort traffic are separated by priority. The network only admits a certain amount of controlled load traffic to ensure that best-effort service is not completely preempted. Within the controlled load class, subclasses with different delay bounds are provided, separated by priority. To clear bursts, a higher-priority class can temporarily borrow bandwidth from a lower-priority class. Thus, priorities allow sharing of bandwidth in one direction and isolation in the other direction. Within each subclass, the overall delay is minimized by simple FIFO scheduling. Flows within a subclass should all have similar traffic characteristics, so when there is a burst in one flow, the other flows can share the delay without being too much delayed themselves. Figure 2.4 illustrates an example of this hierarchical approach.

2.3 RSVP protocol overview

The *Resource ReSerVation Protocol* RSVP ([ZDE+93]) was jointly developed at the Information Sciences Institute of the University of California (ISI) and Xerox Corp.'s Palo Alto Research Center (PARC). Development is carried on in the RSVP and Integrated Services working groups in the IETF (Internet Engineering Task Force). The RSVP specification [BZB+97] and some accompanying documents will advance to Proposed Standard status in the near future.

RSVP is used to set up reservations for network resources. It communicates a request for QoS along the data path, resulting in a reservation if the request was successful. A common misunderstanding is that RSVP alone provides better quality of service. RSVP is a control protocol that sets up a reservation, but enforcement of the reservation needs to be done by another component of the architecture. It is like making a travel reservation. The booking system makes sure a seat is available on every leg of the journey, then marks the seat unavailable for everyone else. However, if there is nobody at the airport to check that only passengers with reservations (and a few stand-by passengers) are allowed on the plane, a reservation will probably not help to get to one's destination in time.

Each node capable of resource reservation has several modules that work together

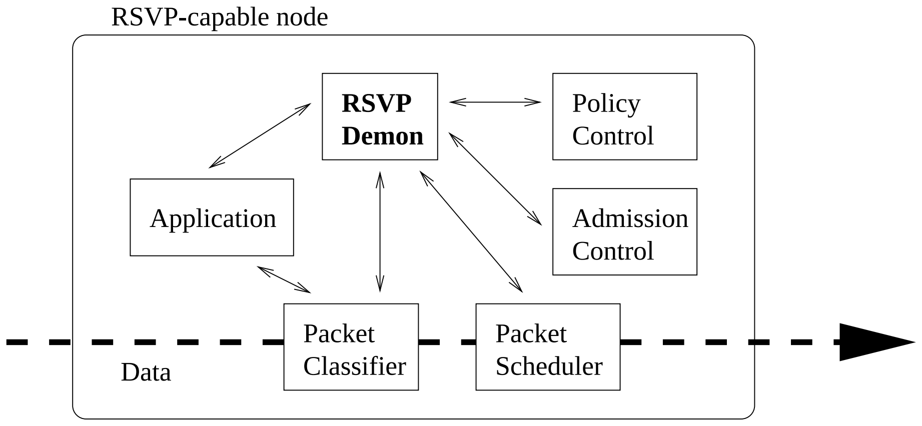

Figure 2.5: Interaction between modules on an RSVP capable node

for reservation setup and enforcement, as illustrated in Figure 2.5. The Integrated Services architecture of routers and hosts is the same, but the implementation usually is different. An *RSVP demon* on every node handles all protocol messages needed to set up and tear down reservations. The RSVP specification [BZB+97] specifies the RSVP demon. Requirements for the other modules are specified elsewhere ([Her97], [BKS97] etc.).

An application requests a certain quality of service from the RSVP demon running on the host. The demon checks with a *policy control* module to see if the user has administrative permission to make a reservation. Future accounting for reservations will also be done by policy control. Policy control determines who can make a reservation. It is subject of continuing research. Open issues are authentication, access control and accounting. The RSVP demon also checks with an *admission control* module to find out whether the node has sufficient resources to supply the requested QoS. Admission control has to find a balance between over-subscribing the link and under-utilizing the bandwidth. One approach is presented in [JDSZ95]. If both checks succeed, parameters are set in other modules to enforce the reservation. The RSVP demon then sends the reservation request to the next hop on the data path. The admission control there also checks if the node can support the desired QoS. The policy control module checks for permission. If either check fails, the RSVP demon sends an error notification back to the host. Otherwise, the parameters for the reservation are set and the reservation request is passed on. As soon as the reservation is accepted by every node on the data path, the flow should receive the requested quality of service.

The *classifier* and *packet scheduler* modules on every node are responsible for the quality of service given to a flow for which a reservation has been made. The classifier looks at every data packet to determine whether the appropriate flow has a reservation and which QoS the flow should get. Flows are identified in one of several formats. The simplest format contains the sender IP address and sender port together with the destination address and port. In this case, the classifier has to look at the IP and UDP/TCP headers. The packet scheduler then makes the forwarding decision according to the QoS class. For example, the packet scheduler decides in which queue to put the packet. The packet scheduler is a key component of the architecture because it actually gives different services to different flows. To ensure that flows receive their requested quality of service, the packet schedulers on all nodes must support the distinction between different services. A packet scheduler could be implemented as described in Section 2.2.3.

At least at the border of the network, routers need to check whether flows conform to their traffic specifications. This is called traffic policing. Violations are handled by traffic reshaping or by tagging or dropping non-conforming packets.

2.3.1 RSVP design features

Receiver-initiated reservations

RSVP makes resource reservations for both unicast and multicast applications. If a sender had to maintain a reservation for each receiver, the protocol would not scale for large multicast groups. Therefore, reservations are receiver-initiated. A sender does not need to know the number and specifics of reservations or the location of the receivers. In this respect, RSVP closely follows the Internet multicast model. Also, a receiver often knows best which quality of service is needed. Different receivers might request and receive different QoS.

To make a reservation, a receiver sends a reservation request (*"resv"*) message towards the source. Figure 2.6 shows the components of resv messages with two examples. Reservation requests specify the requested service (*Rspec*) and the size of the expected data flow (*Tspec*). The *filterspec* specifies which packets can use the reservation. There are three reservation styles: *fixed filter* (FF), *shared explicit* (SE) and *wildcard filter* (WF). In fixed filter style, a reservation is made for packets sent by exactly one sender

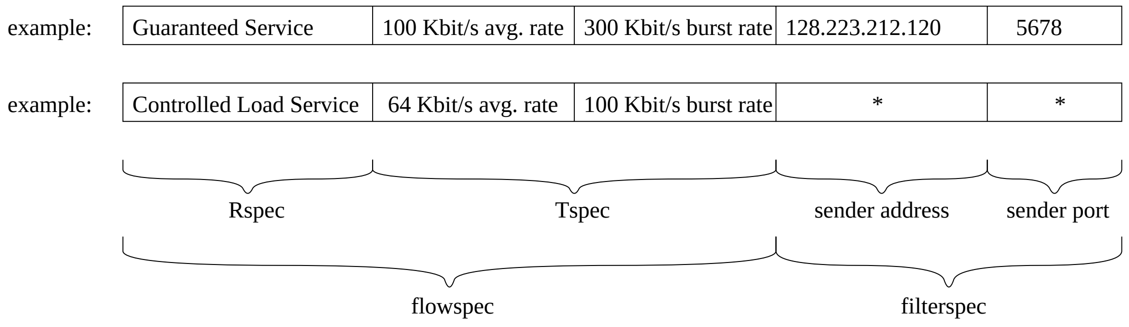

Figure 2.6: Examples of reservation requests

that is specified in the filterspec. In shared-explicit style, packets from several sources listed explicitly in the filterspec can use the reservation. In wildcard style, all sources sending to the multicast group address share the reservation. The second example in Figure 2.6 contains a wildcard filter. Shared and wildcard filters are useful for applications that are self-limiting in their bandwidth needs. These include audio sessions, because usually not more than one or two participants talk at the same time.

Each receiver makes a reservation according to its own needs. At the branch points in the multicast tree, the reservations from different receivers are merged so that only the larger reservation is passed on towards the sender. Thus, RSVP scales well with the number of receivers in a multicast session.

The disadvantage of receiver-initiated reservations is that a receiver does not know which path the data packets are taking. The data path between sender and receiver might be different from the route from the receiver to the sender. To solve this problem, a sender sends *path messages* to the unicast or multicast destination address of the data. A path message contains the unicast address of the next RSVP-capable host upstream (towards the sender). A node that receives a path message saves this information as path state. It then passes the path message on with its own unicast address as the next upstream RSVP hop. Path messages build a trail of path state on the data path. The path state is used to route reservation requests upstream. Resv messages are routed hop-by-hop (unicast) to the next upstream RSVP node.

Path messages also contain information about the sender (as a filterspec) and about traffic characteristics of the data flow that the sender will generate (a Tspec). Receivers use this information to choose an appropriate size for a reservation request. Path messages may also carry *advertising information* (an *Adspec*) which contains QoS information that is updated on every RSVP node. Receivers may then decide whether the advertised QoS is sufficient for their purposes. The error terms for guaranteed service that are exported and collected by the network are an example of the use of advertising information.

Transparent non-RSVP clouds

To allow incremental RSVP deployment in the Internet, RSVP was designed to work transparently across non-RSVP clouds. No explicit tunneling is necessary. Non-RSVP routers route path messages towards the receivers like any other unicast or multicast packet. Path state is set up only in RSVP capable routers. Resv messages are sent as unicast packets from one RSVP-capable node to the next RSVP-hop upstream. Non-RSVP capable routers in the path do not affect the operation of the RSVP protocol. There is no traffic control in non-RSVP nodes, however, so the end-to-end QoS is unpredictable.

Soft state

RSVP needs to function correctly even when control messages are lost or hosts lose their entire state information in a crash.

Soft state is state information that needs to be refreshed periodically, otherwise it times out. Path and resv messages are periodically sent at a configurable refresh interval. State is modified by simply sending the new state. Reservations that are no longer needed should be torn down. If end-systems are unable to do so, the path and reservation state will time out eventually.

Short refresh periods increase control traffic overhead. Long refresh and time-out periods lead to unused capacity that is reserved but no longer needed.

Independence from routing protocols

RSVP is not itself a routing protocol, but it depends on existing unicast and multicast routing protocols. RSVP needs to look up routes and obtain notification of route changes. The routing interface of RSVP is subject of continuing research because more advanced routing features are desirable. One issue is route pinning. Currently, a route is changed when a better route is found, even though the old route is still functioning. A flow that had a reservation on the old path might be unable to get a reservation on the new route, or it might be unwilling to risk a gap in service just because the new route is a few hops shorter. Flows should have the option to continue to use their old path.

RSVP would also take advantage of QoS-based routing, which is an entire research area. Flows should get a route according to their desired QoS. This is a very hard problem, since the QoS on a path changes dynamically.

2.3.2 Applicability as intended by the RSVP working group

RSVP is designed to set up reservations for individual application flows. The overhead of setting up state along the data path is justified only for long-lasting data streams. The average rate of the data flow should be significantly larger than the rate of the control traffic. Reservations are meant for applications that need one of the services defined by the Integrated Services working group, for example controlled load or guaranteed quality of service.

Integrated Services is not intended to improve performance for elastic applications with short-lived connections like telnet, ftp or Web access. Their performance is improved more efficiently by providing adequate capacity and good congestion control.

The RSVP applicability statement [MBB+97] briefly describes uses of RSVP that are currently feasible and identifies areas of limitations. Deployment of RSVP is encouraged in intranets where access control, scalability and security are not critical issues.

Chapter 3

Quality of service in the current Internet model

In this chapter, we describe several experiments that provide insight into the causes of jitter and packet loss in the current Internet model. We draw conclusions about the ability of Integrated Services mechanisms to control and to guarantee quality of service.

3.1 Measuring quality of service parameters

The playback quality of a real-time application is determined by packet loss and jitter. The *packet loss rate* is the fraction of all packets that do not arrive at all. It is straightforward to measure packet loss, since every RTP packet has a sequence number. *Jitter* is not as clearly defined. There are definitions relating to average and maximum jitter. Some authors understand jitter as the difference between the longest and the shortest delay in some period of time. Others define jitter as the maximum delay difference between two consecutive packets in some period of time. Because we want to examine jitter over a period of time, measuring maximum jitter is not meaningful. Instead, we chose a third definition that is standardized in the specification of RTCP in [SCFJ96]. According to this definition, jitter is a smoothed, averaged function of the delay differences between consecutive packets over time.

In order to measure delay variation, we look at the difference in transit times between two consecutive packets. In general, sender and receiver clocks are not synchronized,

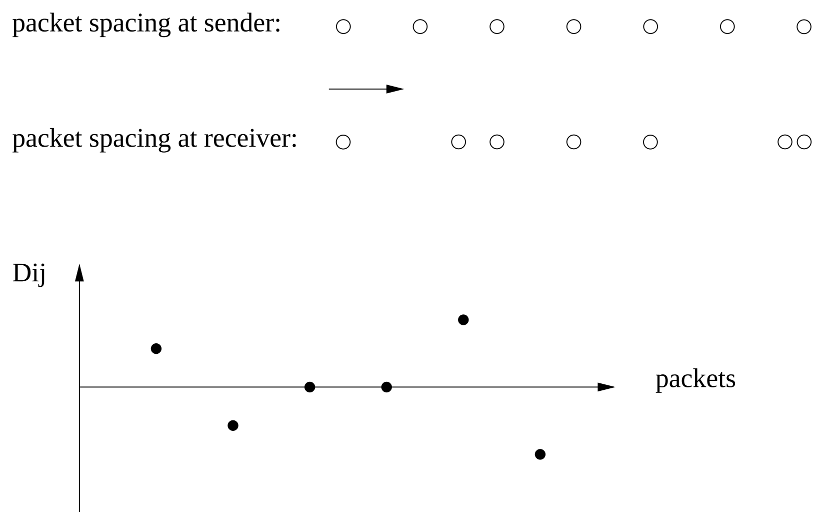

Figure 3.1: Irregular arrivals and resulting delay differences

but they time-stamp packets in the same unit. Although the absolute transit times are unknown, the differences in packet transit times are independent of absolute clock values. If packets i and j are time-stamped with timestamps S_i and S_j when they are sent and are received at times R_i and R_j, respectively, then $D_{ij} = (R_j - S_j) - (R_i - S_i) = (R_j - R_i) - (S_j - S_i)$ is the difference in transit times in timestamp units. Different encodings have different timestamp units, as described later in this section. Measurement tools must take the different timestamp units into account. The sending timestamp S_i is approximated by the timestamp in the RTP header. R_i is obtained by getting the local time immediately after the packet is received.

Usually, we are interested in the delay difference between consecutive packets, thus $j = i + 1$. If packets arrive at exactly the same rate they were sent, D_{ij} is zero. For a packet that is late relative to the previous packet, D_{ij} is positive. If the next packet arrives in time, that is with average delay, D_{ij} is negative for this pair of packets. Thus, one late packet leads to two D_{ij} values unequal to zero. Figure 3.1 shows an example of seven packets that are sent out at equal time intervals but arrive irregularly. Note that the absolute network delays are irrelevant. The D_{ij} values only describe the relative differences in delay.

According to the definition in the RTCP specification, jitter is a smoothed function of $|D_{i-1,i}|$:

$$J_i = J_{i-1} + (|D_{i-1,i}| - J_{i-1})/16 = 15/16 * J_{i-1} + 1/16 * |D_{i-1,i}|$$

This function is a simple example of a class of estimators called "recursive prediction error" or "stochastic gradient algorithms" ([Pos81], [Jac88]).[1] Here, we are not making predictions, but we aggregate the $|D_{i-1,i}|$ observations into a stable estimate of the most recent jitter. Thus, jitter is a sort of running average of all $|D_{i-1,i}|$ from the beginning of the measurement up to the current packet. Recent packets have a larger weight than older packets. Our measurements show that the jitter curve needs about 100 packets to stabilize (see for example Figure 3.5).

Lost packets are simply ignored in the jitter calculation. They could be understood as packets with infinite delay, but that would be impractical for the calculation. Since lost packets don't have an effect on the playback point, including them in the jitter measurement has no benefits.

Actual sending times can only be approximated by RTP timestamps. In order to get measurements that only include network delays, not waiting times in the sending hosts, the (multicast) test flow needs to be received at two hosts on the same path. Both receivers log all incoming packets and their arrival times. The packet sequence number then serves as a key to determine which packets arrived at both receivers. From these packets, the jitter that was introduced between the two receivers is calculated. Thus, the jitter originating on arbitrary subpaths can be measured. We call this the "differential jitter" method. It is described in more detail together with the experiments in Section 3.2.3.

The difference between audio and video flows

Audio sources sample sound at 8000 Hz. Each sample is encoded in one byte or less, if compression is used. Packets containing a constant number of samples, for example 160, 320, 640 or 1280, are then sent out every 20, 40, 80 or 160 ms. The timestamp clock ticks at the sample rate of 8000 Hz. Timestamps of consecutive packets are exactly the

[1]In the TCP protocol, the estimator is used in the form $A \leftarrow (1-g)A + gM$ to predict round trip times, where g is a gain factor, A the prediction and M the new measurement.

number of samples in a packet apart. Therefore, audio flows have a constant packet rate and each packet has a different RTP timestamp.

The timestamp clocks of Internet video encoders usually tick at a rate of 90000 Hz. Video encodings are often differential, so the packet rate and the size of the packets depend on changes in the image. Several consecutive packets may have equal RTP timestamps if they are logically created at once, e.g. belong to the same video frame. Obviously, we would measure jitter that isn't real if we calculated delay differences between all consecutive pairs of packets. Therefore, only the first in a series of packets with equal timestamps can be used for jitter measurement. On the downside, video jitter measurements do not reflect the delay of every packet and thus lose in accuracy.

3.2 Causes of jitter

3.2.1 Bursty traffic and queues

Most of the work on quality of service guarantees implicitly assumes that queuing is the one most important cause of delay variation. This section explains how queuing leads to jitter. The following sections deal with other causes of jitter that should not be overlooked.

If all packets of a flow encounter the same queues and queue lengths on the path, they all wait for about the same time. The end-to-end delay may be high, but there is no delay variance. Jitter comes into play when consecutive packets experience different waiting periods in the queues. If packet scheduling is done in strict first-in first-out manner, a difference in delay means that a queue grew or shrunk between arrivals of two consecutive packets. As more complicated scheduling and queuing mechanisms become widely used, however, this might not necessarily be true. From the position of a packet in a queue it can no longer be concluded how long it has been sitting in the queue or when it will be serviced.

Queues build up in a switch or router whenever the input rate is larger than the capacity of the output link. Dynamic queues can result from only one flow. Imagine a bursty flow that traverses a serial line (or a fraction of a serial line) of limited capacity. If the burst rate of the flow is higher than the capacity of the line, a queue builds up for

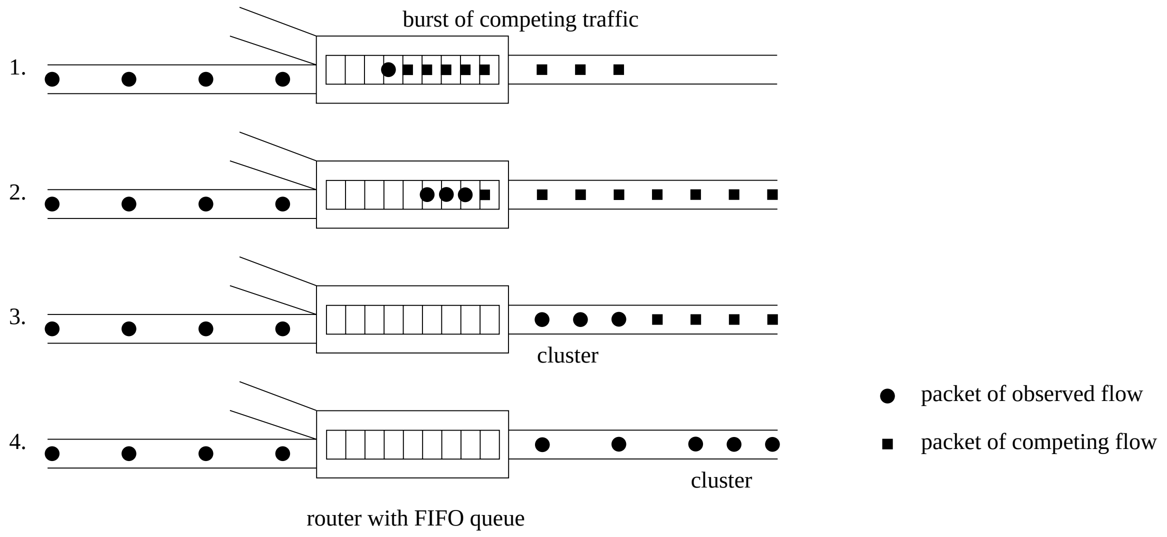
burst of competing traffic
1.
2.
3.
cluster
4.
cluster
router with FIFO queue
● packet of observed flow
■ packet of competing flow

Figure 3.2: Clustering of packets

the duration of the burst. A packet later in the burst experiences a longer delay than an earlier packet, thus there is jitter. Queues often result from two or more flows competing for the same output interface, however. If the combined momentary incoming packet rate is larger than the bandwidth of the outgoing link or the processing speed of the interface, packets are queued.

Under the theoretical assumption that queuing is the only cause of delay variations, the maximum difference in delay between two consecutive packets is bounded in the case of strict FIFO queuing. It is determined by the sum of the maximum queue lengths on the path. This also means that the maximum delay variation can grow linearly with the number of hops in the path, however. The worst case occurs when one packet is not queued at all and the next one finds all queues full. A more likely scenario is that bursts of competing flows lead to *clustering of packets* in a flow. This situation is illustrated in Figure 3.2. Consider several bursty flows competing for bandwidth at different hops in the path. At hops where the outgoing bandwidth is less than the combined incoming bandwidth, another flow's burst might be ahead in line. The next packet might arrive while the first one is still queued (part 2 of Figure 3.2). These two packets are then sent out very shortly after another on the outgoing interface (parts 3 and 4 of Figure 3.2). They arrive together at all subsequent hops, where the same can happen again. At each hop, there is a possibility that the cluster is further delayed, leading to a maximum delay variation that increases with the number of hops. In the case of complex packet scheduling with several prioritized queues, certain flows might not be serviced temporarily, which leads to unbounded delay variation.

The maximum delay variation might also be rather small on a long path. Experience shows that queue waiting periods in independent queues partially cancel each other out, so each packet experiences some average delay ([Jac94]). Finally, delay is high but jitter is low when the switches work under a constant overload and their queues are always close to full.

The relationship between end-to-end delay variation and the length of the path is difficult to predict. Under certain circumstances, the maximum delay variation increases linearly with the number of hops in the path. Under different circumstances, the delay variation is independent from the number of hops. Since nothing can be done about the length of a path, applications that can't tolerate much jitter have to use a service that limits the total time spent in the queues.

3.2.2 Experiments: Host-induced jitter

Sending hosts do not always behave as expected, especially if they have more than one task to do. If audio or video hard- and software is involved in unexpected behavior, applications at the other end might see jitter even though the network is lightly loaded.

The timestamp in an RTP packet usually denotes the time of sampling and thus indicates when the content of the packet should be played in relation to other packets. The point in time when the packet was actually sent is unknown. Nevertheless, this timestamp is often used to compute "network delay" or "network jitter". If a considerable amount of time passes between sampling and sending, these measurements are wrong. The delay or jitter is real, since applications have to deal with delay variations between sampling and playback. However, limited conclusions can be drawn about the state of the network.

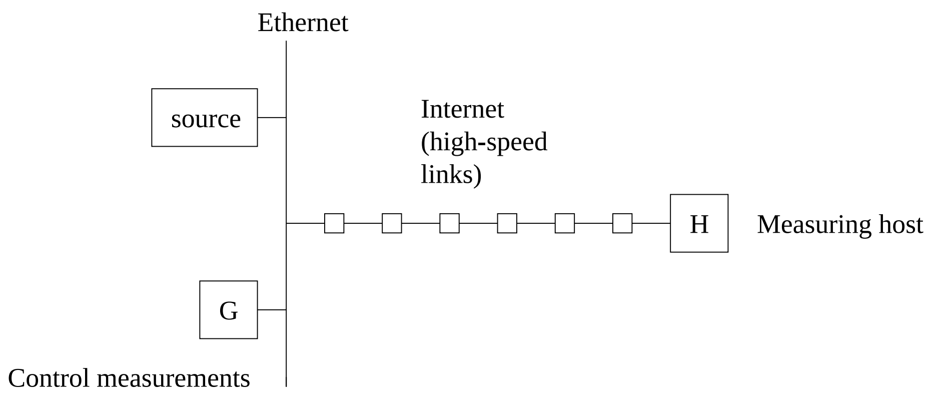

Figure 3.3: The test environment to measure host-induced jitter

Jitter as a scheduling artifact

We observed sources with unexpected behavior in various parts of the MBone. One of these sources is connected to the receiver via a few hops on a lightly loaded high-speed network, as shown in Figure 3.3. Nevertheless, we receive audio packets two or three at a time at host H. This jitter did not seem to be caused in the network, but by the encoding host. We eliminated the possibility of a strangely behaving network element by measuring arrival times at host G in the uncongested LAN of the sender. Usually, audio sources send packets at a constant rate. In this case, the source sends 25 packets per second. The packets should arrive approximately 40 ms apart. From this source, however, two or three packets are received back to back every 80 ms. The difference in delay between consecutive packets is thus almost the 40 ms.

In Figure 3.4, arrival times are plotted over RTP timestamps. Regular arrivals would plot as a straight line, because the encoding used by this source generates audio packets at a constant rate. But here you can see groups of two and three packets arriving at almost the same time. Using the formula for recent average jitter in Section 3.1, the jitter is a fairly constant 35 ms. Relative to this huge jitter, delay variations due to queuing in the network are insignificant. In the plot, the network-induced jitter is completely masked by the host-induced jitter. Figure 3.5 shows the differences in delay between two consecutive packets and the resulting jitter.

One explanation for this host behavior is a scheduling artifact. The host spends most of its cycles encoding video. The audio process doesn't get to run often enough to send out packets as soon as they are sampled.

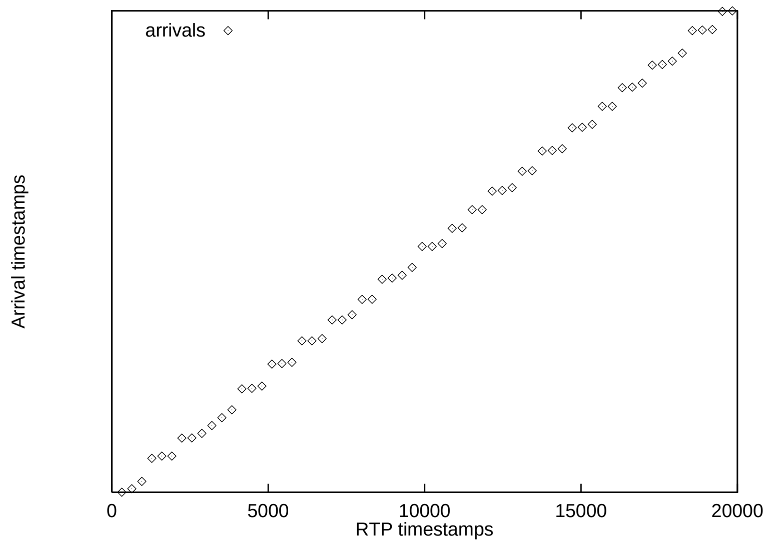

Figure 3.4: Arrivals with host-induced jitter

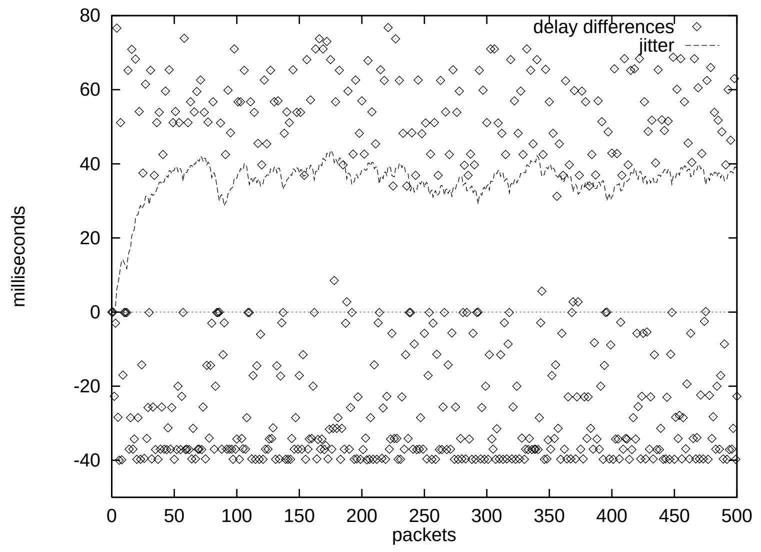

Figure 3.5: Host-induced delay variations and jitter

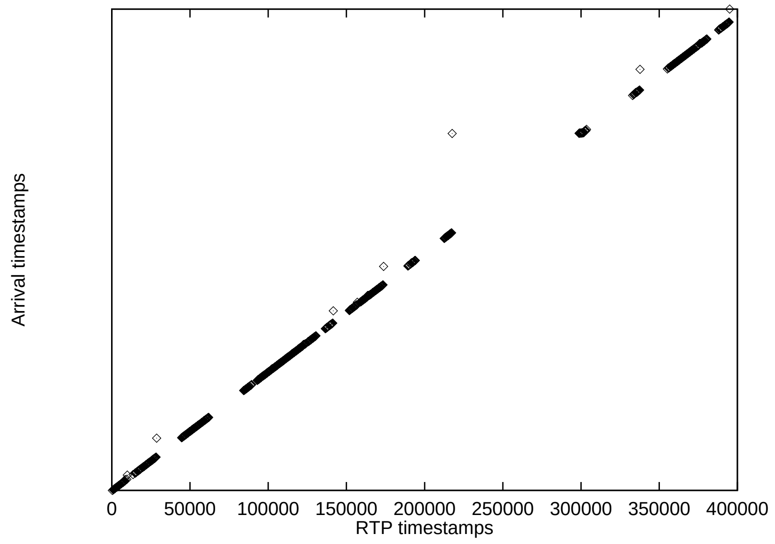

Figure 3.6: Arrivals of “leftover” packets after gaps

Jitter in the context of silence suppression

We discovered another form of host-induced jitter in an audio flow that shows delay variations of several seconds. Except for these huge jitter spikes, the flow behaves normally. Packet loss rates and overall jitter are low. A closer look at the data reveals that the jitter spikes are associated with gaps in the transmission. The observed flow originates at an audio-only source somewhere in the MBone.

Figure 3.6 shows a plot of arrival times at the receiver. The gaps are periods when the source sends no data. Sometimes the last packet before a gap is delayed for the duration of the gap. It arrives just ahead of the first packet after the gap. It is highly unlikely that a such a delay is caused by a network element, since packets are not held in routers for several seconds until more packets of the same flow flush them out. Here, these “leftover” packets are obviously held in the outgoing interface of the sending host for the duration of the transmission gap. They are then sent out as soon as the sender starts transmitting again. This behavior is probably due to a scheduling problem in the operating system of the sending host.

Figure 3.7 shows the delay differences and the resulting jitter as measured at the receiver. Again, even an unloaded network would not improve the end-to-end quality in this case. The negative components of the delay differences are not shown, because they exactly mirror the positive components.

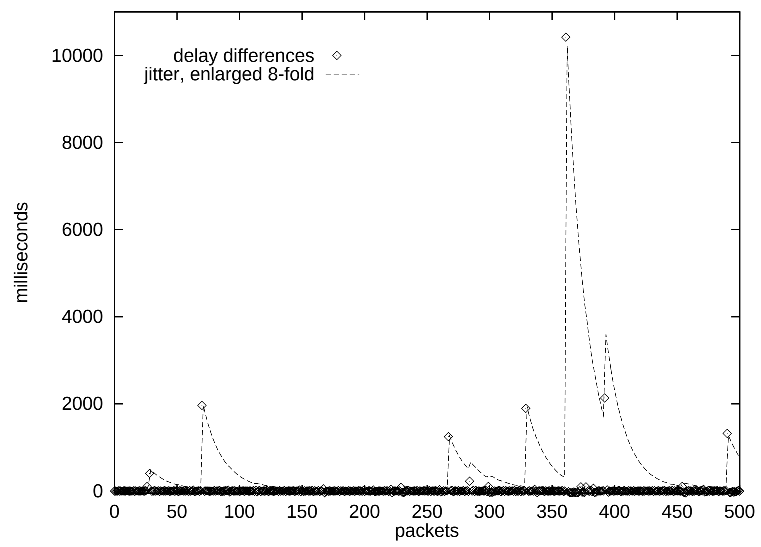

Figure 3.7: Delay variations and jitter caused by "leftover" packets

A packet that misses its playback point by several seconds is simply discarded by the receiving application. However, packets that arrive only a fraction of a second late might cause the receiver to adapt its playback point to accommodate these packets. Thus, playback quality is better but the total delay is larger. This is an example of the trade-off between interactivity and fidelity. Gaps in audio transmissions are commonly caused by silence suppression at the source application. In the absence of an audio signal, audio applications don't source packets. This behavior is usually controlled by a "suppress silence" option in the application. Silent periods are especially frequent in lectures and conversations. For lectures, long delays are not a problem. For conversations, maybe a simple fix can be applied at a sender that shows this behavior: turn silence suppression off. This fix sacrifices some bandwidth that is used to transmit silence, but it avoids "leftover" packets and the problems associated with them.

Other hosts introduce a less significant amount of jitter when they suppress silence. We observed that in several cases, the first packet after each silent period is a little late. The arrivals in Figure 3.8 were measured by a receiver on the LAN of the sender (see Figure 3.9). This eliminates the possibility of network jitter caused by routers. In theory, the delays could be caused by CSMA/CD on the LAN, by strange receiver behavior or by the source. Because of the pattern in the delays, it seems unlikely that LAN access difficulties cause the jitter. Because the same receiver also measured flows

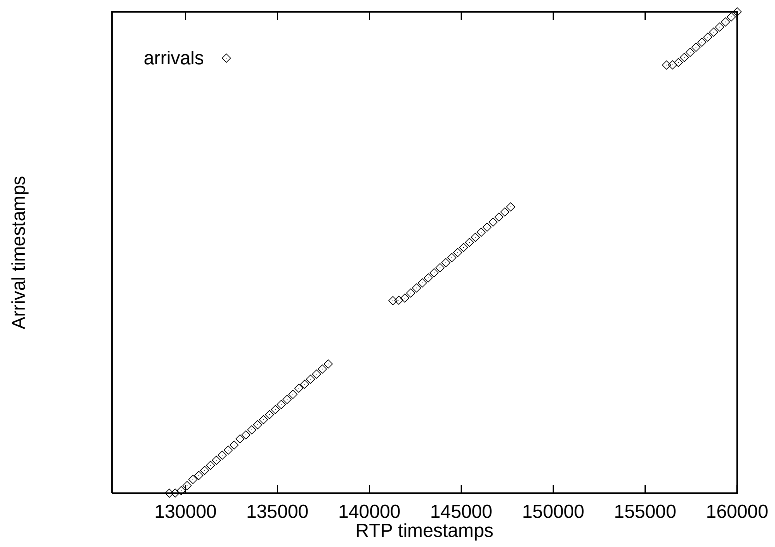

Figure 3.8: Arrivals after silent periods

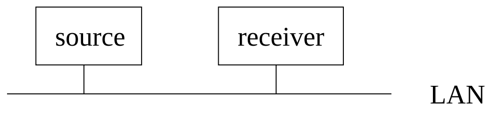

Figure 3.9: Test setup for local silence suppression measurements

that did not show delayed packets after transmission gaps, it also seems unlikely that the receiver causes the jitter. Therefore, we believe that this observation is another example of host-induced jitter.

3.2.3 Experiments: Router-induced jitter

In an experiment originally designed to quantify the effects of FIFO queuing on jitter, we found that the jitter caused by queues is small compared to the jitter that is introduced by a router when fast switching is turned off.

Figure 3.10 shows our test environment. The test flows are the only traffic in the test environment. An audio source sends an audio flow that is received at host H. The goal is to measure only the jitter introduced by routers R1 and R2. To make sure sender-induced jitter or jitter introduced by other routers on the path does not distort the results, host G also receives the audio flow. We then use the "differential jitter" measurement method described in Section 3.1. Both hosts log RTP timestamps and arrival times of the incoming packets. Corresponding RTP timestamps are then used to

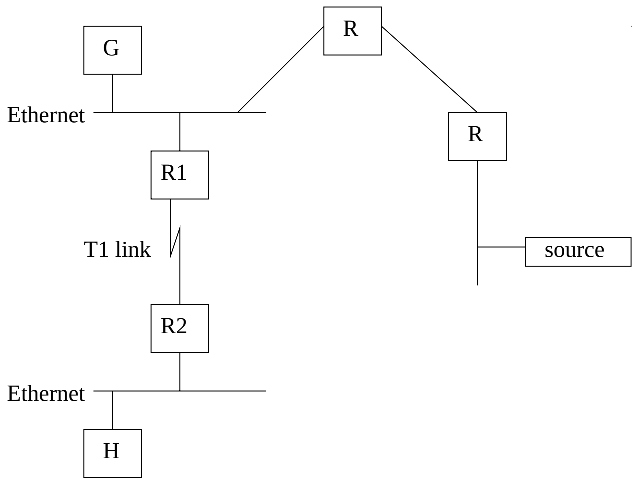

Figure 3.10: Test environment to measure router-induced jitter

calculate differences in delay and jitter between G and H. This is done according to the formulas in Section 3.1 for delay differences and jitter. However, to calculate only the "differential" jitter between G and H, arrival times at G are used as sending times to compute the difference in delay. Thus, the difference in delay D_{ij} between two packets i and j is the time it takes packet j to go from G to H less the time it takes packet i to go from G to H. $R_{k,G}$ ($R_{k,H}$) is the arrival time of packet k at G (H).

$$D_{ij} = (R_{j,H} - R_{j,G}) - (R_{i,H} - R_{i,G}) = (R_{j,H} - R_{i,H}) - (R_{j,G} - R_{i,G})$$

Jitter is a smoothed function of $|D_{i-1,i}|$, as defined in Section 3.1.

Routers R1 and R2 are cisco 2503 routers, a low-end model that is widely used in private networks and in the Internet. To measure the effect of FIFO queuing on jitter, queues need to be generated in the router without putting too much load on the surrounding system. To achieve this, we limited the output capacity of router R1 by using a T1-link run at 512 Kbit/s as a bottleneck on the path to H. Under high load, queues build up in the outgoing interface of R1, while R2 forwards packets onto an unloaded ethernet without queuing.

We need to ensure that the experiment setup itself does not cause jitter. One concern is the use of only a fractional T1 link. A full T1 connection contains twenty-four time division multiplexed 64 Kbit/s channels for a total bandwidth of 1.536 Mbit/s (e.g. [Dow96]). In rotation, each channel gets an 8-bit time slice. Since we are only using

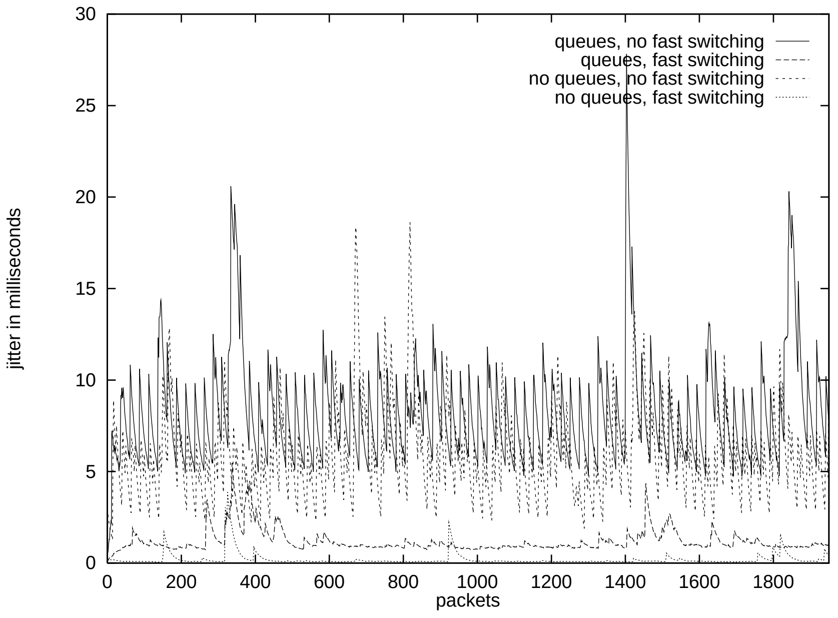

Figure 3.11: Jitter of multicast traffic

512 Kbit/s, sixteen channels are unused. If the unused channels are scheduled in one block, the interface cannot transmit for up to sixteen 8-bit time slots. This corresponds to $(8 * 16/15360000)s = 83\mu s$. Fortunately, this maximum possible jitter of 83μs is negligible compared to the jitter of several milliseconds that we are observing.

Another concern is the burstiness of the flow to be measured. If the burst rate of the flow is higher than the bandwidth of the serial link, packets are queued in R1 although the network is otherwise idle. Thus, burstiness of the test flow needs to be avoided. Additional control measurements at G confirmed that packets arrive at G at approximately constant time intervals.

In the first part of the experiment, the source sends to a multicast group that hosts G and H join. First, we looked at the jitter introduced by R1 and R2 without additional load on them. As expected, the jitter was close to zero except for a few spikes. Then we used a traffic generator to build up queues in R1. At 535 64-byte packets per second of additional multicast traffic flowing through R1 and R2, the router showed snapshots of the queue length between 0 and 39 out of 40. 0.5% of the audio packets in the test flow were dropped. The varying length of the queue led to a jitter of about 1.25 ms. See the two bottom plots in Figure 3.11.

We then turned off the multicast route cache in both routers and repeated the ex-

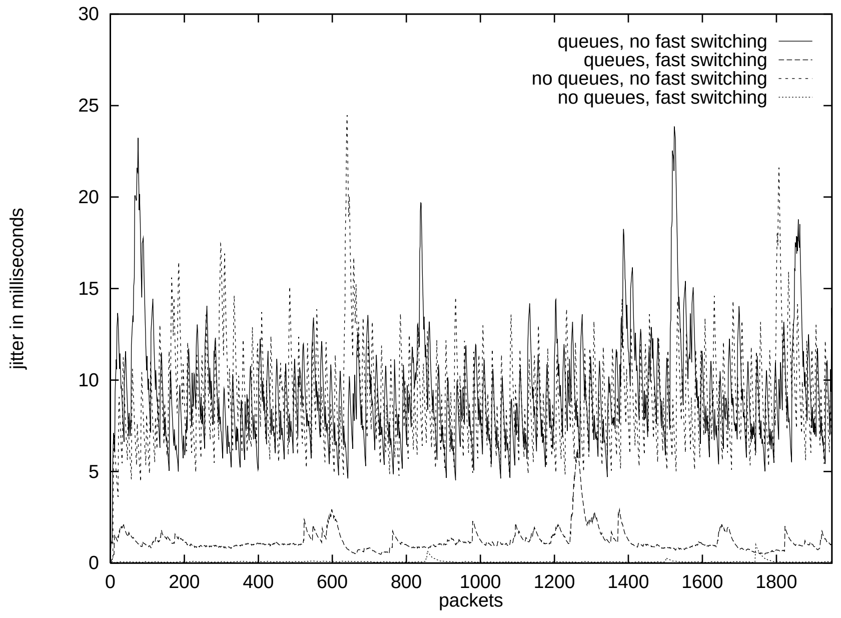

Figure 3.12: Jitter of unicast traffic

periment. Without a multicast route cache, routers are forced to make the routing decision for each packet by looking up the destination in the multicast routing table. Surprisingly, we observed very high jitter when the cache was not used. Without additional load on the routers, the jitter was about 5 ms. With queues in R1 it was about 7.5 ms.

These results are essentially the same for unicast traffic, as shown in Figure 3.12. When the IP route cache is used and there are no queues, the jitter is virtually zero. With queues built up by additional unicast traffic, the jitter is still only 1.3 ms. Without using the cache however, the jitter is about 9 ms, no matter if there are queues in the router.

Since a routing table look-up should take the same amount of time for packets with the same destination, the cause of the jitter is not obvious. But a close look at the arrival times of the packets at H reveals a regular pattern. However, the pattern is difficult to see in the arrival plot in Figure 3.13 because of the large range of the timestamps. Figure 3.14 uses a trick to make the pattern visible. Only the 1000 least significant milliseconds of all arrival timestamps are plotted. There are 25 packets per second that are plotted in a very steep line. Because irregularities occur at whole second intervals, this way of plotting reveals the pattern. Figure 3.15 clarifies the structure of Figure

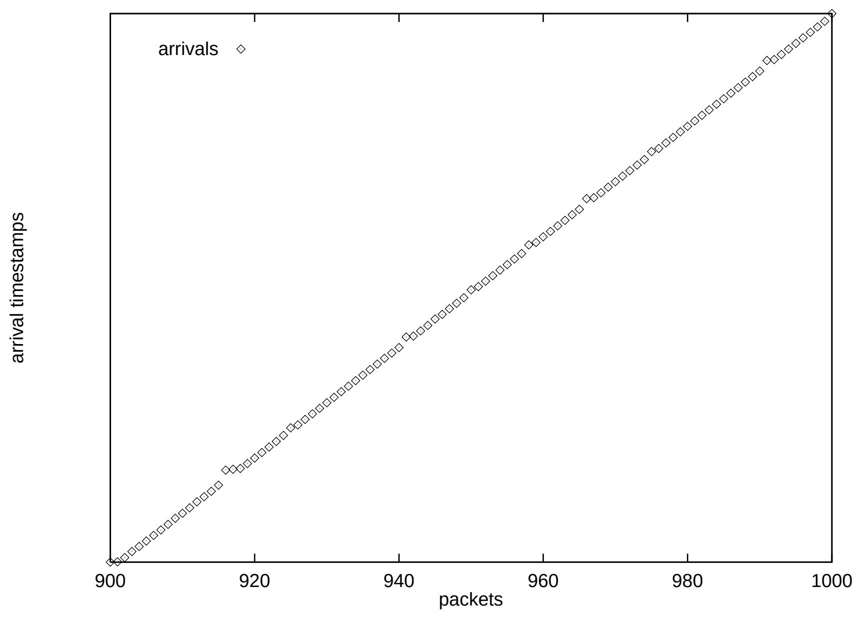

Figure 3.13: Arrival timestamps with fast switching turned off

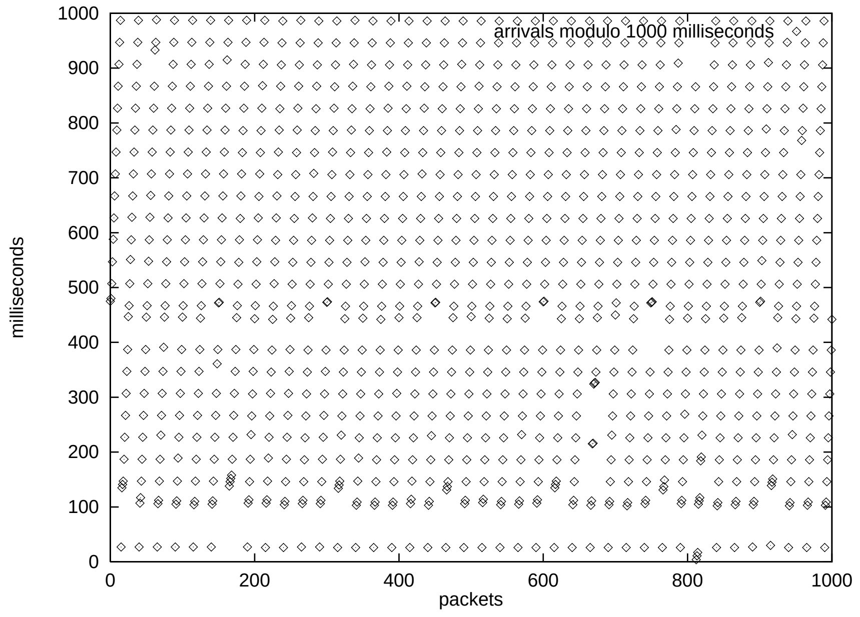

Figure 3.14: Delay pattern with fast switching turned off

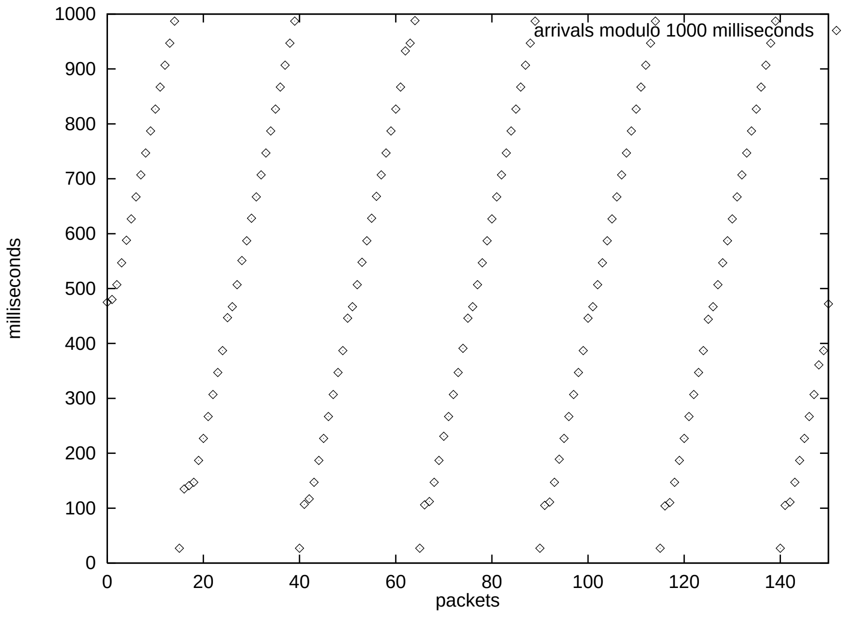

Figure 3.15: Clarification of the plotting trick used for Figure 3.14

3.14 by plotting only a small number of packets.

Each router produces its share of the jitter. The two empty-looking rows at 100 and 400 milliseconds correspond to the two routers. We verified this with an experiment in which fast switching was turned off in only one router. Each router delays one packet per second by 40 ms or 20 ms. Once every six seconds, each router has a gap in its service that leads to a cluster of two or three packets, thus delaying a packet by 80 ms. Both routers seem to periodically be doing something that keeps them from forwarding packets in a timely fashion. We can only guess what they are doing. When the caches are turned off, the main CPU is involved in the forwarding of every packet. If the main CPU has another large task to do, the service is interrupted and packets are buffered. Periodic tasks could include garbage collection, consistency checks and routing updates.

A delay variation of up to 80 ms every few second does not seem to be a serious problem. But we haven't encountered the worst case yet. If the packet that is delayed in the second router happens to be the one that also was delayed in the first router, delay variations quickly add up to unacceptable delays. The worst-case delay variation is the sum of the maximum delay variations in each router. Also, compared to possible queuing delays, a delay variance of 80 ms in one router is considerable. At a data rate of 9.2 Kbyte/s that is broken into 25 equal-sized packets per second, it takes 5.75 ms to send one packet on a 512 Kbit/s link. Thus, a delay variation of 80 ms corresponds to a queue variation of thirteen packets. On faster interfaces, a gap in service of about 80 ms has the same effect as a sudden build-up of an even larger queue.

In every-day operation, there is no reason to turn fast switching off. However, this is the mode where one would expect less irregularities because of the absence of caching effects. For debugging purposes, it is important that the non-optimized method works as expected.

The observations described before are specific to one router architecture and to specific hardware and software. The results can not be generalized. However, these observations serve as an example of the unpredictability of network elements.

3.2.4 Implications of jitter not induced by queuing

All jitter sources discussed in the previous sections cause jitter unintentionally. These forms of jitter are most likely due to problems in the implementation, or to the interference of several components, or to situations that weren't anticipated. Although they should not exist, they are real, and applications have to deal with them.

A problem arises for applications that need guarantees on jitter bounds. The specification of guaranteed quality of service by the Integrated Services Working Group ([SPG97]) requires that each network element exports per-hop *error terms*. Error terms specify how the element's implementation of the guaranteed service deviates from the fluid model. There are rate-dependent and rate-independent error terms. An example for a rate-dependent error term is the time it takes to reassemble a datagram from ATM cells. The maximum amount of time a packet might need to wait for its assigned slot in a slotted network qualifies as a rate-independent error term. Another example are gaps in the service of a router because of the processing of routing updates.

Three aspects are worth highlighting. First, the error terms reflect worst-case scenarios. Applied to the case in which turning off caching in the routers led to periodic delay increases, this means that every router on the path has to export its worst-case delay. The total error term is the sum of the per-hop error terms, which can get quite large. It is the nature of a guarantee that although it is rather unlikely for a packet to experience worst-case delays in every router, this small possibility needs to be accounted for. Therefore, the guaranteed jitter bound will be so high that it is arguable whether the service will be of any use.

Second, exporting error terms requires that they are known. Accounting for serialization of ATM cells or waiting periods for a slot seems straightforward, but is isn't quite as obvious why the absence of fast switching would lead to delay variations. The draft on guaranteed service assumes that all error terms can be derived analytically. This approach involves the risk that unexpected sources of jitter are not accounted for. As a result, a service might be advertised that can't support the given guarantee, which is probably worse than if no guarantee had been given in the first place.

Last, the error terms as specified in the draft do not include host-induced delay variations. In order to compute the maximum datagram queuing delay, the end nodes need to include bounds on the host-induced jitter in the calculation. Again, the end node has to know about this jitter. Also, there might not be a bound, as in the case of jitter in the context of silence suppression.

While unexpected jitter sources can lead to a violation of the service contract in guaranteed service, the effects on controlled load service are not quite as bad. The user gets what he or she asks for: service as in a lightly loaded network. The service might not be good, but applications that use controlled load service can usually tolerate some amount of jitter and an occasional lost packet. However, it does not make much sense to make a reservation and use controlled load service when a significant part of the jitter is not network jitter. The user is interested in end-to-end quality of service, but controlled load service can only influence part of the end-to-end jitter.

The existence of significant jitter that is not caused by queuing makes it impossible to use jitter as an early indicator for congestion. Misinterpreting jitter might lead to unnecessary adaptations or under-utilized links.

The only way to prevent jitter caused by irregularities in hosts and routers is to use real-time operating systems, which are a topic of current research.

3.3 Causes of packet loss

IP networks are unreliable in nature. Some packets just disappear, but most "lost packets" are consciously discarded for various reasons. One reason that is becoming less common in modern fiber networks is damage to the packet due to a transmission error. If the checksum check fails, the packet is simply discarded. Packets with an undefined format are also dropped.

By far more often, packets are dropped because of congestion. If there is not enough buffer space in the routers, queue overflows occur. A router usually has incoming interface buffers, system buffers and outgoing interface buffers. Depending on where a packet is dropped, the drop is called an input drop or an output drop. Input drops usually occur when the router can't process packets fast enough, while output drops occur when the outgoing link is too busy.

Routers also try to avoid congestion by dropping packets before the queues have reached their maximum length. This mechanism is called *random early detection* (RED, [Jac88]). It causes TCP sources to back off and slow-start, thus temporarily reducing the amount of traffic going through the router. However, as the UDP portion in Internet traffic becomes larger, this mechanism becomes less effective and unfair, since only TCP sources back off.

3.3.1 Effects of packet loss in audio and video applications

When packets carrying video data are lost, the video application can't update the current frame. The image may become inconsistent (e.g. moving elements appear twice) or the video image changes abruptly when the next packets arrive. However, limitations on video encodings for low-bandwidth links make it difficult to distinguish effects resulting from these limitations from effects due to losses. In audio applications, packet loss leads to choppy sound. Crackles and gaps in the replayed signal make speech difficult to understand and music less than enjoyable.

In an informal experiment, we tested the effects of various levels of packet loss on intelligibility. Figure 3.16 shows the test setup. As a test flow, we used a news channel that is transmitted over the MBone by a nearby source. Without intervention, there was no packet loss. We then used a traffic generator to overload the link between routers R1 and R2, forcing the router to drop packets. Even a packet loss rate of

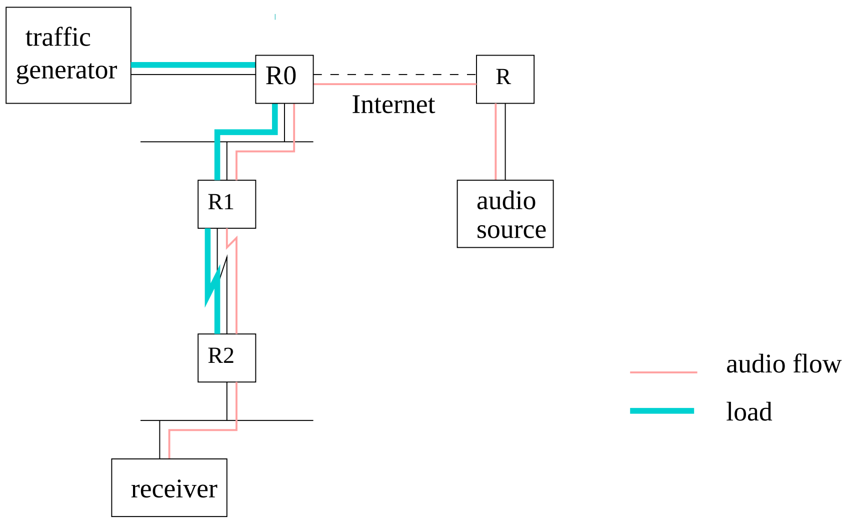

Figure 3.16: Test setup for intelligibility experiments

1% is clearly noticeable as a crackle. At up to 13% packet loss, all words can be understood, but there are a lot of crackles. At 15%, a news anchor is understandable, but it requires a lot of concentration to follow an interview. At 20%, most sentences are still understandable thanks to the redundancy in human languages. Non-redundant information like numbers can get lost. It is impossible to understand speakers with a strong accent. At 25% packet loss, only parts of phrases are understandable. For most people, this renders a transmission useless.

These results are intended to provide an estimate on what to expect from a flow with a certain packet loss rate. Of course, intelligibility depends on many factors, especially on the quality of the original signal. Lower packet loss can be tolerated in audio conferences or interviews than in a clear news broadcast. The quality of the audio hardware and of speakers or headsets plays a role. Finally, the number of audio packets per second influences intelligibility. At only six packets per second, the loss of one packet can mean the loss of half a phrase. At fifty packets per second, one lost packet is a crackle. Thus, when packet losses are rare, intelligibility can be improved by distributing the risk of loss over many smaller packets.

3.4 MBone traffic characterization

In order to design a better service model for the Internet, one needs to know the weaknesses of the present model. The current best-effort service causes packets to get dropped, reordered, delayed, corrupted or duplicated. The goal of this section is to examine the characteristics of service for multicast traffic under the current best-effort model.

The Internet Protocol discards corrupted packets, so damaged and dropped packets are not distinguished. The reasons for dropping packets were discussed in the previous section. Duplication can occur while the routing protocol is stabilizing. Often, multicast flows are briefly duplicated because group memberships and thus multicast routes change dynamically. For example, the multicast routing protocol PIM/dense mode ([DEF$^+$97]) causes transient duplication when a new receiver is reachable via two different routers on the same shared LAN. Packet reordering does not seem to be common. Most widely used multicast routing protocols do not support secondary routes for load sharing. Therefore, route changes that affect uninterrupted flows are rare. They only occur when a shorter route is found, in which case packet reordering is possible. In our tests, we did not see any reordered packets except for the moment when a reservation is turned on in a router (see Section 4.2.4).

Packets do get dropped and delayed frequently in the Internet. Packet drop rates can get arbitrarily high. Because IP multicast is not reliable, MBone applications suffer from packet loss. Audio flows with a packet loss rate of more than 30% are completely unintelligible and therefore useless. Video flows with high packet loss rates might be of some use if the image is static. In both cases, it is not worth worrying about jitter. In the first case, it is irrelevant whether a useless flow has high jitter. In the video case, a static video image that can tolerate high packet loss can also tolerate high jitter. Therefore, we are only interested in the jitter of flows that have less than 30% packet loss. The single most effective quality improvement for flows with higher packet loss rates is to ensure they get adequate bandwidth.

Congestion usually leads to packet clustering, as explained in Section 3.2.1. Figure 3.17 shows packet arrivals of an MBone audio flow originating in another part of the US that obviously traversed highly congested networks. The observed flow suffered 31% packet loss. Groups of up to eight packets arrive at almost the same time, with big

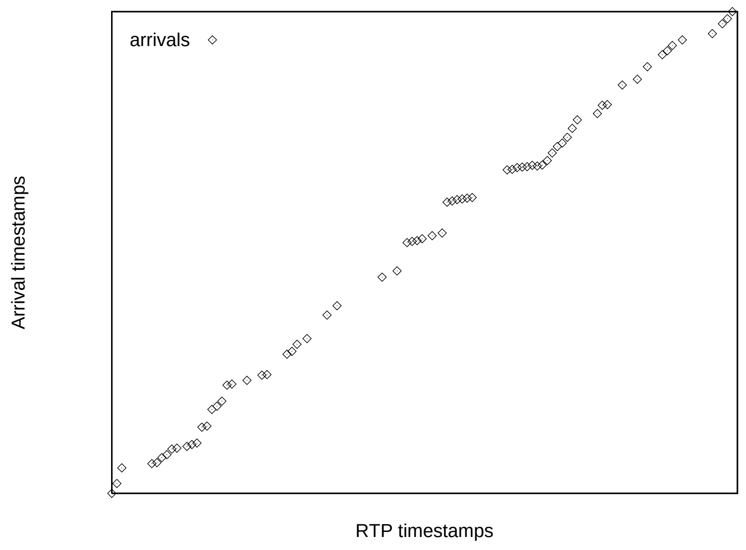

Figure 3.17: A distorted MBone audio flow

gaps between groups. Less severe clustering is shown in Figure 3.18. This MBone flow also originated somewhere in the United States. It was replayed faithfully by vat except for the crackle caused by about 2.5% packet loss. Occasionally, groups of two or three packets arrive together. The delay differences and the resulting jitter are plotted in Figure 3.19. To accommodate delayed packets, the playback point has to be moved by about 130ms. In jitter plots for video flows, clustering isn't as prevailing. This does not mean packets aren't clustered, however. Since only the first packet of each frame is considered for the delay calculation, jitter plots for video flows are not detailed enough to see this effect.

Intuitively, one would expect that flows with high packet loss rate also have high jitter. This is certainly true for very badly congested paths. However, there is no relationship between packet loss and delay variation for "useful" flows, that are flows with a loss rate of much less than 30%. Figure 3.20 shows the results of an experiment that measured loss rates and delay variations of various MBone audio flows. We took two 500-packet samples from each flow. Samples were taken about ten minutes apart. Each sample is represented by two points, plotted over the loss rate. The upper point is the maximum delay difference encountered in the sample. The lower point is the 90% quantile. This means that 90% of delay differences in the sample are less than the value of the lower point. Most of the lower points are grouped in pairs. A pair represents two samples of one audio flow. This shows that network conditions were stable over a ten

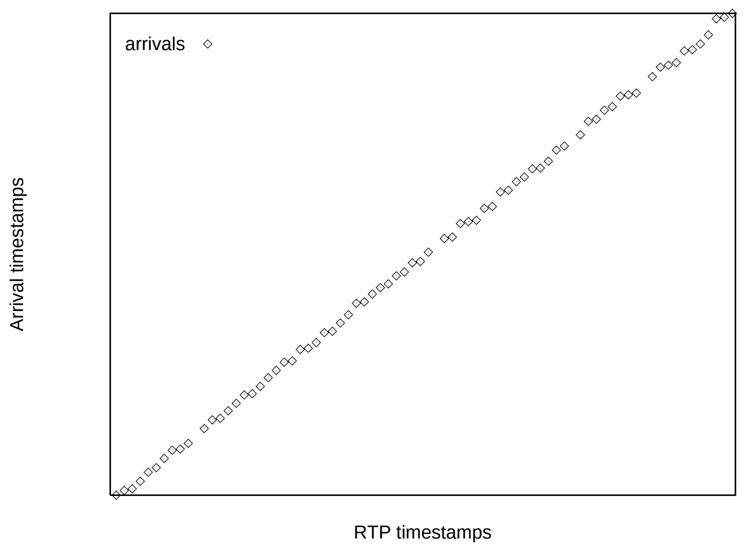

Figure 3.18: Clustered arrivals in a good-quality audio flow

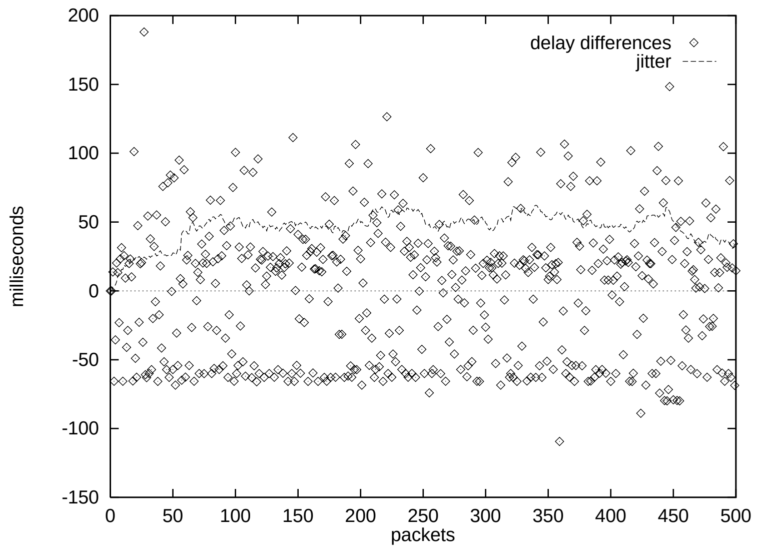

Figure 3.19: Jitter and delay differences of a good-quality audio flow

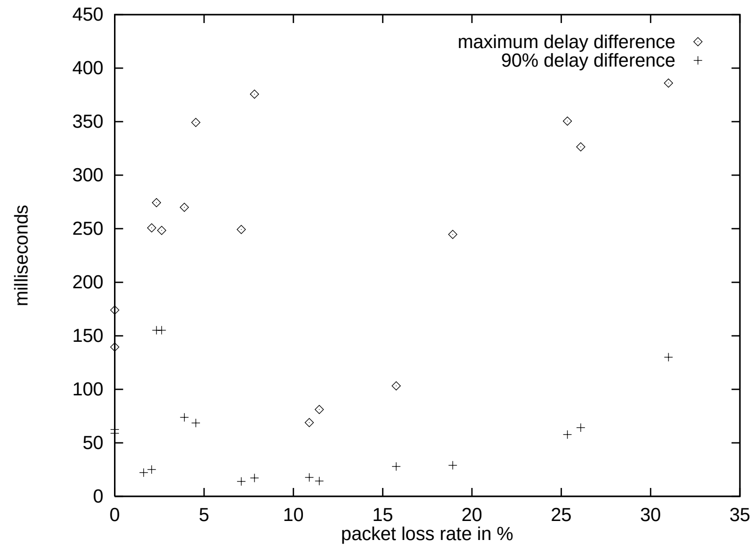

Figure 3.20: The (non-)relationship between packet loss and delay

minute period of time. Packet loss rates and the 90% quantile stayed about the same except for one flow where the packet loss rate changed from 16% to 19%. The fact that two samples of the same flow have very similar 90% quantiles shows that these values are meaningful, whereas the maximum delay differences seem fairly random.

There is no positive or negative trend in the data in Figure 3.20. That means, flows with high packet loss rates do not necessarily have high jitter in the packets that do arrive. Flows with low packet loss rates might have high jitter. Considering the various causes for jitter discussed in Section 3.2, this is not surprising. However, let's assume queuing was the only cause of jitter for a moment. As explained in Section 3.2.1, it is not the absolute length of the queues that leads to jitter, it is their growing and shrinking. Thus, bursty flows that are competing for output links are responsible for delay variations independent from long-term congestion. Even if the average bandwidth is sufficient, there might be significant jitter, caused by bursts. As a consequence, improving interactivity by reducing jitter might be worthwhile in some cases even if the packet loss rate is acceptable. On the other hand, one does not know whether the delay variations in Figure 3.20 are in fact queuing jitter or whether they are partially caused by hosts and misbehaving routers.

The absolute values of the delay variation suggest that although it is rather low in most cases, interactive applications could have problems with delay variation, especially if they can only tolerate loss rates of a few percent.

3.5 Summary of Experiments

Experiment	Observation	Conclusion
Host-induced jitter caused by a scheduling artifact (Section 3.2.2)	Packets are not sent out in regular time intervals, but in groups.	Controlling network jitter does not necessarily lead to good end-to-end quality. Real-time operating systems are necessary to prevent host-induced jitter.
Host-induced jitter in the context of silence suppression (Section 3.2.2)	I. Packets are held in the outgoing interface for the duration of the transmission gap.	
	II. The first packet after a transmission gap is sent late.	
Router-induced jitter (Section 3.2.3)	When fast switching is turned off, routers periodically delay packets significantly.	Unexpected jitter sources make guarantees impossible.
Perception of packet loss in audio applications (Section 3.3.1)	All packet loss is audible. Speech is understandable at up to 13-15% packet loss. 25% packet loss renders a transmission useless.	
Measurements of various MBone audio flows (Section 3.4)	I. Irregularities in arrival times are often in the form of packet clusters.	Congestion leads to packet clustering.
	II. There is no strong correlation between the packet loss rate and the jitter of a flow.	Sufficient average bandwidth does not guarantee good end-to-end quality.

Chapter 4

Evaluation of Integrated Services and RSVP

This chapter discusses characteristics and problems of RSVP. Experiments in a controlled environment show the effect of reservations on packet loss and jitter. We also discuss conceptual difficulties concerning QoS on the link layer, quantitative guarantees and scalability.

4.1 General discussion of RSVP

It is still controversial whether Integrated Services and RSVP are good approaches to provide quality of service for real-time applications on the Internet. The Integrated Services working group argues that special services and predictable quality of service are necessary for the class of playback applications ([BCS94]). Without restrictions on the amount of traffic in a datagram network, real-time flows suffer from congestion, leading to unacceptable packet loss and jitter. Reservations ensure that the network does not accept more traffic than it can handle and applications receive their desired quality of service. Others believe that the congestion problem can be solved with adaptive applications and hierarchical encodings alone ([Jac94], [CDF$^+$95]). Receivers can adjust their playback point to deal with varying delay. During congestion, a rate-adaptive sender might switch to a less expensive encoding. Applications might temporarily confine themselves to the low-resolution levels of a hierarchically encoded data stream. However, all real-time applications need a certain minimum bandwidth and quality to be useful at all. In the present-day Internet, such a minimum bandwidth is not always available.

If there is no congestion, resource reservation is not worth the effort. Buffering is a more effective way to remove the small amount of jitter that is caused by queuing in an uncongested network. Van Jacobson estimates in [Jac95] that a buffer of only 800 bytes is needed to remove jitter from a transcontinental voice conversation.

Some people believe that at some point in the future, bandwidth will stop being a problem. So far, however, the bandwidth needs of applications have grown steadily with the available network capacity. This trend will continue unless applications have some incentive not to be greedy. Accounting and billing for reservations could be a way to provide that incentive. When bandwidth is free or billed for as a flat rate, resources are often wasted. Per-flow accounting and billing would reduce the demand on network resources to the amount that is really useful for users. Furthermore, demand could be smoothed over time by differentiated fee structures. Opponents argue that accounting will never be feasible on a per-flow basis across the Internet. The amount of data would be unmanageable and there are too many parties involved. The solution that Internet providers would prefer is to offer reservations only in cases where sender and receiver subscribe to the same provider.

A service offering a certain QoS is only acceptable if reservation requests are successful most of the times. Steve Deering argues that therefore, enough bandwidth needs to be available to meet normal peak demand ([CDF+95]). Reservations would then be unnecessary. Under the assumption that detailed billing can decrease and shift demand, this argument does not hold true, however. Moreover, the necessary bandwidth does not exist yet. RSVP could be an incentive to invest in bandwidth where it pays off. On the other hand, it might even be more profitable to invest in bandwidth without supporting the overhead of RSVP (see Section 6.1).

Internet purists oppose any state in the Internet, soft or hard. They argue that reliability suffers and overall performance deteriorates because of the overhead. One factor that has allowed the Internet to grow to its current size is that it requires relatively little state in intermediate hops. RSVP requires a considerable amount of state, thus limits scalability. RSVP has a set-up phase to convey parameters (and state) to the network, a conversation phase and a tear-down phase. This closely resembles the call model in a telephone network, which is considered a step backwards.

The Integrated Services architecture is designed to be general and flexible. Implementations are therefore complicated and put much strain on routers. So far, RSVP is manageable and safe only in smaller private networks, as pointed out by the Integrated Services working group in [MBB$^+$97]. In these networks, it might be easier to simply add bandwidth than to deploy RSVP. On the other hand, Integrated Services shouldn't be abandoned just because they are not completely feasible with today's technology. There are some fundamental problems, however. RSVP does not scale well with the number of sessions, as will be shown in Section 4.5.

Another fundamental issue is the notion of quantitative QoS guarantees. Integrated Services are designed for IP networks, but IP networks are unreliable. It is not part of the system model for any network element to guarantee anything. Especially delays can't be completely controlled by services on top of IP. Section 4.4 explains in more detail why quantitative guarantees don't make sense.

RSVP approximates a solution to a distributed scheduling problem. Distributed scheduling problems are NP-hard, if an optimal solution is to be found. Many network engineers believe that a simpler approach should be favored.

Finally, there is the social debate. Opinions are divided about the desirable behavior of a network during overload. Either everybody's service is degraded (as done by traditional best-effort service) or some flows get good service and others get very bad service (which is what reservations will do). Who should be allowed to make reservations? With reservations, users aren't equal any more. It is arguable whether reservations make the Internet more fair or less fair. It seems that capitalism will not spare the Internet. Users who are willing to pay for better service will get better service, either through RSVP or through other forms of differentiated service.

4.2 Experiments: Reservations on small interfaces

4.2.1 Experiment setup

The test environment in Figure 4.1 is used to measure the effect of reservations on packet loss and jitter in a small, controlled environment. It is the same hardware setup as for the experiments on router-induced jitter. The routers are low-end cisco models that are widely used in the Internet.

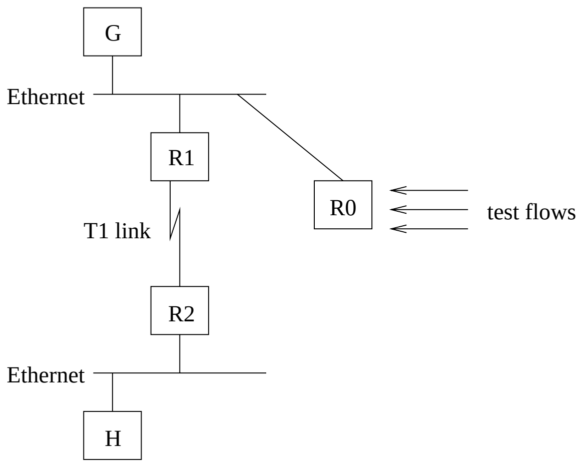

Figure 4.1: Test environment for static reservations

We used the traffic generator MGEN ([Ada])to generate multicast test flows. MGEN allows to specify the number of packets per second and their size. It then generates packets at a constant rate. MGEN can thus generate flows with exactly the same characteristics as audio flows. Host H joins the respective multicast groups, so the test flows go through routers R1 and R2 and the serial link between them. There is enough bandwidth in the system for the test flows except for the bottleneck between routers R1 and R2.

We do not run an RSVP demon on H or on the senders. Therefore, reservations are set up only for the serial line, not for the end-to-end path. For this purpose, Cisco routers provide static reservations that can be configured into routers. Enforcement of static reservations in the routers is the same as for end-to-end reservations. There are two parts to a static reservation: an “ip rsvp sender” configuration and an “ip rsvp reservation” configuration. In the router closer to the sender (here G), path state is set up manually with the “ip rsvp sender” configuration. This causes the router to behave as if it just received a path message. The router then sends path messages to establish path state on the router(s) downstream. On the router closer to the receiver (here H), a reservation is manually configured with the “ip rsvp reservation” configuration. This causes the router to behave as if it just received a reservation request (“resv”) message. Upon receipt of path messages, resv messages are sent towards the sender, establishing reservations on the upstream router(s). A downstream router that receives a path message has no way of knowing whether the message originated from a static reservation or from the RSVP demon at the source. The same holds true for resv messages. Upstream routers don't know whether the resv message originated from a static reservation or from the RSVP demon at the receiver.

Reservations are set up only in those two routers that have state manually configured and in all routers between them (here, only in R1 and R2). This is true for the following reasons. If a resv message is sent further upstream from the router that has set up static path state, it is simply ignored at the next router that does not have path state ([BZ96], here at R0). Path state further downstream from the router with the static reservation is set up if the routers or hosts are RSVP capable. (In our experiment setup, there are no RSVP capable routers downstream from R2.) Without a corresponding reservation, path state does not have any effect, however.

Currently, cisco routers support controlled load and guaranteed rate service. However, the current implementation doesn't seem to distinguish between those. Fair queuing ([CSZ92]) has to be in effect in the routers to enforce the reservations. In this implementation, every flow gets its own queue, even best-effort flows. A higher priority is assigned to reserved flows, but controlled load and guaranteed service reservations get the same priority. Thus, both kinds of reservations have exactly the same effects in the following experiments.

4.2.2 The effect of reservations on packet loss

To determine whether flows for which reservations have been made really get their reserved bandwidth, we conducted several experiments with a varying number of test flows (between two and twenty). All test flows are generated by MGEN and have a constant bandwidth. The total bandwidth of all flows combined is exactly the capacity of the serial link between routers R1 and R2. Nevertheless, the offered load slightly overloads the serial link. Router R1 operates under a constant overload and thus drops a total of 2% of all packets going through R1. This total loss rate of 2% is independent of the number of reservations, which we varied between zero and fifteen. Reservations determine which flow's packets are dropped. Once reservations are in place for some flows, these flows have a packet loss rate of zero, while the flows without reservations share the total loss of the link. For example, when half of the bandwidth is taken by reserved flows, flows without a reservation experience a packet loss rate of 4% while reserved flows have no packet loss. The overhead for reservation management, queue management, packet classification and scheduling for up to 20 flows does not lead to increased total packet loss compared to first-in first-out scheduling. The limiting factor in this experiment setup is the bandwidth of the serial link, not the processing speed of the router. Thus, the overhead does not reduce the performance of the router. Flows with reservations fully receive their reserved bandwidth. If reserved flows conform to their traffic specification, they enjoy zero packet loss. We conclude that reservations seem to work very well in providing certain flows with their necessary bandwidth on small interfaces.

However, it is worth mentioning that implementation errors in the router software can easily break the no-loss guarantees given by controlled load and guaranteed service. In our experiments, routers needed to be occasionally rebooted because they seemed to have a memory leak and started dropping packets from reserved flows after a series of experiments.

4.2.3 The effect of reservations on jitter

To compare the jitter of a test flow with and without a reservation, we needed a bursty load. Two hosts behind router R0 generate bursty multicast flows. Each of these flows periodically switches between high (150 and 160 packets/s) and low (10 packets/s) packet transmission rates. The periods are different, so the total load periodically varies in a larger range. Packet sizes and rates are different for each flow to avoid synchronization effects. The bursty flows together with the test flow almost completely fill the serial link but do not overload it. The ethernet interface on router R2 joins the multicast groups to which the bursty flows are sent. Thus, there is a bursty load on router R1 and the serial link, but not on host H. This avoids distortion of the measurement at host H due to too much load on host H. The flow that is measured closely resembles an audio flow with a constant rate of 25 packets per second. It is received at hosts G and H. The delay variations and jitter originating between G and H are calculated as described for router-induced jitter on page 32.

The top plot in Figure 4.2 shows the jitter of the test flow with simple FIFO scheduling in R1 and R2. During the first 2200 packets, the bursty flows also compete for the

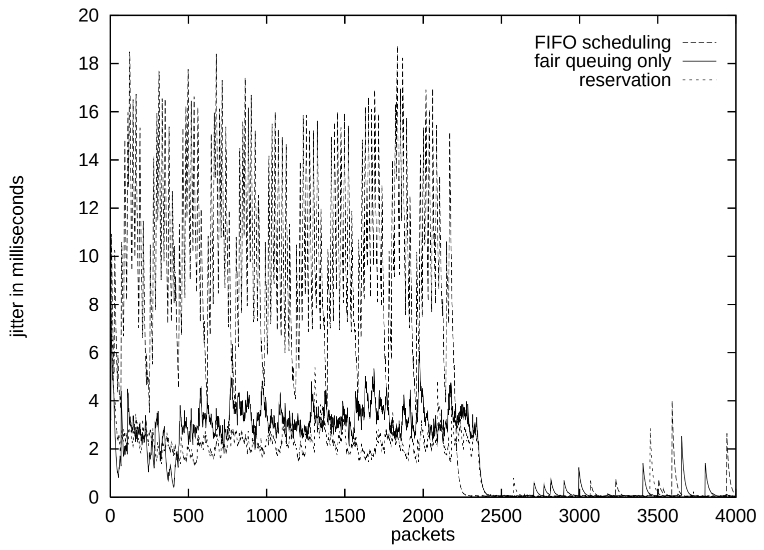

Figure 4.2: Jitter of an audio flow with bursty competing traffic

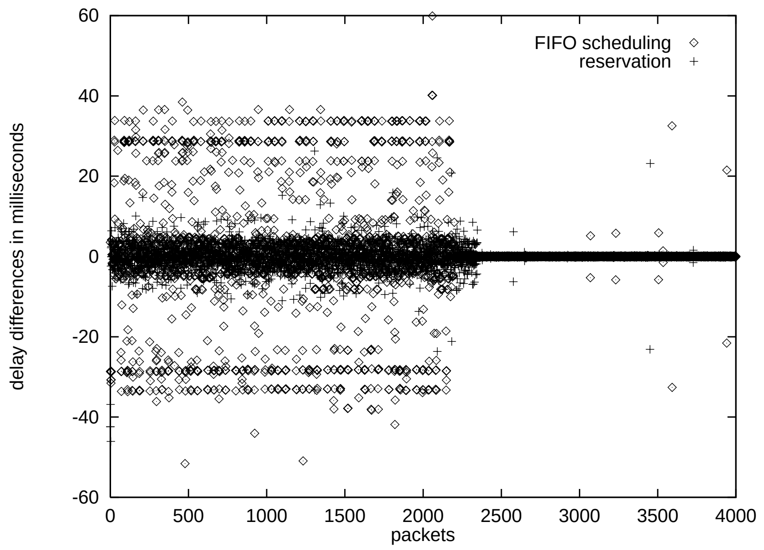

Figure 4.3: Delay variation with bursty competing traffic

serial link. The jitter is periodic because the bursts are periodic. We then made a static reservation on the serial link for the test flow, which also requires fair queuing. The bottom plot in Figure 4.2 shows the resulting jitter. It is much lower than without a reservation, but still higher than when the network is unloaded. Since the test flow itself is not bursty, it should not have jitter. Thus, this jitter needs to be exported as an error term for guaranteed service.

The middle plot shows the jitter without a reservation, but with fair queuing as a scheduling strategy. The current implementation of fair queuing doesn't have weights, so even without reservations, the smaller flows get all the bandwidth they need and the larger flows are disadvantaged. With this implementation, the quality of service that a flow receives depends only on its bandwidth relative to the other flows. Obviously, this is not a good basis for comparison. Therefore, the benefits of a reservation for a flow can only be measured by comparison to the FIFO case.

Figure 4.3 shows the corresponding delay differences. Without a reservation, delay variations of up to 35 ms are common. With a reservation, only about 0.5% of all packets are more than 12 ms late or early. These few larger delay differences are probably not due to congestion, since there are some late packets even after the bursty load is turned off.

In this experiment, a reservation brought the highest common delay variation from 35 ms down to 12 ms. However, the service is not as good as if there was no load on the routers, in which case the delay variation is between 0 and 1 ms. Also, it is unrealistic to assume that by controlling queuing delays, all delay variation is controlled and 100% of all packets will arrive in time.

4.2.4 Effects during reservation setup

RSVP does not provide the possibility to make advance reservations for future sessions. Reservation setup requires path state in intermediate hops, which in turn requires a running sender application. Also, thrifty users will only make a reservation when best-effort service is not satisfactory. This implies that the reservation is set up while the application is running. We found that during a constant overload, it takes several seconds until the reservation is in effect. Moreover, the packets that are in the queue

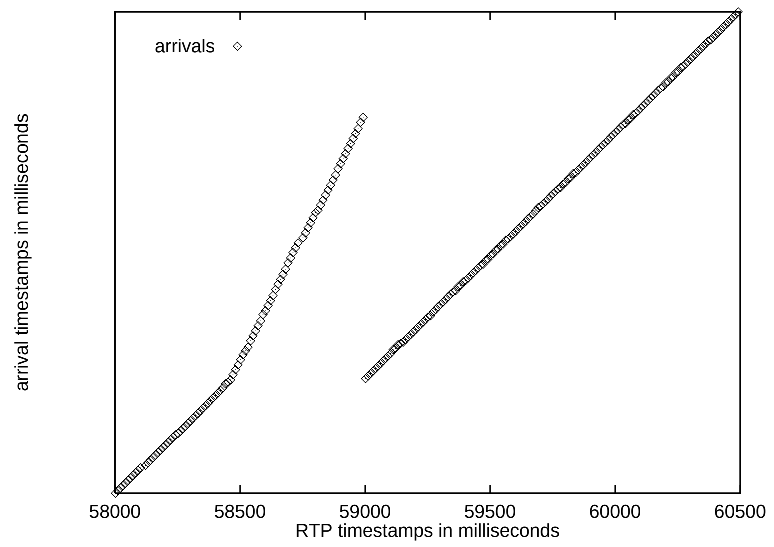

Figure 4.4: Packet reordering during reservation setup

while the reservation is switched on arrive out of order.

In this experiment, we completely filled the 512 Kbit/s link between R1 and R2 (see Figure 4.1) with two multicast flows. The flows have equal data rates, but different packet sizes and packet rates to avoid synchronization effects. We then made a reservation for one of them. Figure 4.4 shows how about 45 packets (those with RTP timestamps between 58500 and 59000) are reordered. They arrive interspersed with later packets, up to one second late. The explanation is as follows. As soon as the reservation is in place, incoming packets bypass the old queues and are sent out immediately. Packets that are still sitting in the old WFQ queue are not discarded, but delayed. The old queue is gradually emptied by interspersing the old packets with the new ones. Thus, one should expect a gap in service while a reservation is set up. The more routers there are on the path that need to reassign their queues, the longer this gap will probably be.

Switching off a reservation has immediate effect. No packets are extraordinarily delayed.

The reservation setup latency is one of the factors that limit the granularity of RSVP flows. Flows need to be long-lived to take advantage of a reservation.

4.3 Lower layers and reservation enforcement

Integrated Services and RSVP are layer 3 mechanisms. RSVP demons run on network layer routers. Integrated Services packet scheduling manages queuing and priorities at the network layer. However, there is little control over queuing and QoS contracts at the link lever. This is a problem at two places: the LAN of the user (the last kilometer between Internet or campus network and the user) and the interior of the network.

4.3.1 Link layer QoS in distribution networks and backbones

The network layer would be able to completely control quality of service if routers were connected by dedicated point-to-point links with the behavior of a wire. But real networks are built from a variety of media and technologies. What looks like a dedicated point-to-point link to a router might actually be just a virtual point-to-point link in a frame relay network. A router knows the capacity of the physical link, but it is not aware of the service contract with the frame-relay network. Each frame relay PVC has a CIR (Committed Information Rate) associated with it. The network will give priority treatment to this amount of bandwidth. Traffic exceeding the CIR is marked "discard-eligible". An Integrated Services router might assume that the entire capacity of the physical link is available and accept reservations based on this assumption. In reality, only the CIR is available with certainty. Consequently, the portion of reserved traffic that exceeds the CIR is likely to be dropped, violating the reservations. Integrated Services routers will need some kind of CIR discovery to avoid this disastrous situation.

Another issue is queuing on the link layer. While routers see dedicated serial links, the reality might be a shared network, such as a frame relay network. Queuing delays in frame relay switches are not under control of any network layer mechanism. Contention is not uncommon in frame relay networks, and the CIR is not a 100% bandwidth guarantee. This means that Integrated Services network elements cannot control all queuing delays and packet losses. They can't even estimate the delays, because they don't see link layer switches. In the case of guarantees service, this will either result in under-estimated total delay estimates (if delay in the frame relay switch is not taken into account) or in over-estimated worst-case estimates (if the worst-case delay of the frame relay switch is taken into account). Controlled load will promise service as in a lightly loaded network, although queuing at frame relay switches can cause significant delay variation.

A similar problem arises with centralized switched technologies like SMDS (Switched Multimegabit Data Service). The way switches deal with temporary overload is queuing. Again, routers don't see this delay and have no way to control or estimate it.

4.3.2 Integrated Services on LANs

Predictable end-to-end QoS requires quality of service on the last kilometer to the user. It is very difficult, if not impossible, to provide QoS on legacy LANs. Often, hosts are connected to the last-hop router through shared ethernets. FDDI is common as a local backbone, with ethernets or token rings attached to the FDDI stations. To differentiate between different kinds of service, a LAN technology must support priorities to isolate reserved and unreserved flows. This requirement rules out all legacy ethernet technologies. It is obvious that shared media with CSMA/CD access protocols cannot provide any service guarantees. CSMA/CD makes it impossible to predict when and how much a station will be able to send. The current trend in ethernet networking is "micro-segmentation", where each host has its own ethernet segment. This increases the bandwidth available for each host. However, without priorities, reserved and unreserved flows cannot be separated. The IEEE is currently working on standards for expedited traffic classes in bridges/switches. The proposed standard requires three priority bits in the ethernet frame header. On shared ethernets with priority, at least some statistical guarantees can be given. To provide deterministic guarantees, ethernet has to be deployed in a switched full duplex topology with priority. This means that there are only two devices on a segment, the host and the bridge/switch, and there is no access contention.

FDDI and token ring offer priorities in their current form. Thus, they have the potential to support QoS guarantees. To use this potential in subnetworks, a signaling mechanism is needed. The ISSLL working group (Integrated Services on Specific Link Layers) in the IETF describes a framework in [GPS97]. In order to provide guarantees, resource reservation has to be done on the link layer. This is in addition to resource reservation on the network layer. Of course, link layer switches need classifiers and schedulers to provide different classes of service. For resource reservation, link layer switches also need to have bandwidth allocators that keep track of reservations. A new protocol is needed so bandwidth allocators can talk to each other. A requester module translates a layer 3 reservation into a layer 2 reservation. It provides the interface between a layer 3 reservation protocol (such as RSVP) and the bandwidth allocator. Although the complexity of link layer resource reservation can be reduced by centralizing part of the mechanism, it still seems to be a considerable overhead.

If legacy technologies are to be used and guarantees are required, there is no alternative to layer 2 resource reservation. ATM to the desktop would make it possible to guarantee QoS, but it is not clear whether ATM to the desktop will become economically feasible in the near future. There are many commercial efforts to provide QoS on LANs. For example, one proposal is to provide an ISDN line for each host parallel to an ethernet.

If guarantees are not required, QoS can be realized with priorities alone. Since LAN bandwidth is relatively inexpensive, it is probably cheaper to add bandwidth than to have a complex reservation mechanism. When plenty of bandwidth is available, a high-priority service without guarantees is just as good as a quantitative guarantee. This will be discussed for general networks (not just LANs) in Section 6.3.

In summary, network administrators will have three possibilities to deal with QoS to the end-user in the future. First, they could buy new hardware that supports QoS guarantees. Second, they could upgrade their LANs to support priorities and deploy the link layer reservation mechanism as proposed by the ISSLL working group. Third, they could over-provision their networks and do without guarantees.

4.4 Fundamental problems with guarantees

Networks provide different levels of guarantees. Strictly speaking, even best-effort service provides a weak guarantee: the network promises not to delay or drop packets unnecessarily. A stronger guarantee is given by simple priority mechanisms. They guarantee that packets with higher priority get better service than packets with lower priority. This is a qualitative guarantee. No commitment is made about absolute end-to-end quality. The actual end-to-end quality depends on the total traffic in the priority classes. Controlled load and guaranteed service attempt to detach quality of service from the current traffic situation. Controlled load service provides only a qualitative guarantee, while guaranteed service attempts to offer strict quantitative end-to-end guarantees.

4.4.1 The service contract of controlled load service

The controlled load provides a service "closely equivalent to unloaded best-effort service" ([Wro97]). To a controlled load flow, the network looks lightly loaded. During normal operation, congestion loss and queuing delays may occur occasionally, but are viewed as statistical effects. This definition of the service guarantee clearly excludes control over jitter effects that are independent of the traffic load. Host-induced jitter and jitter caused by unexpected router behavior (as examined in Section 3.2) fall into this category, so network elements supporting controlled load are not required to limit these forms of jitter. The service guarantee also implies that packet classifying and scheduling must not cause delay variations themselves. Our experiments on the effect of reservations on jitter (Section 4.2.3) suggest that packet scheduling itself doesn't cause jitter in an unloaded network. On a highly loaded router, however, no scheduling algorithm can completely keep up the illusion of the router being lightly loaded. The reason is that packets are scheduled by the packet, not by the bit as in the fluid model. The router may have just started to send a large packet that belongs to another flow when a packet from a controlled load flow comes in. A slow link is then unavailable for several milliseconds, causing delay variation.

In summary, a user of controlled load service can expect the service guarantee to be fulfilled as long as "closely equivalent" is not interpreted too strictly. A realistic number of statistical effects (i.e. packet losses and delays) needs to be allowed so network elements can employ statistical approaches to admission control ([JDSZ95]). Deviation of packet scheduling from the fluid model also needs to be accounted for. Also, a user should be aware that controlled load service does not control all sources of jitter, merely queuing delays in network layer devices.

4.4.2 The service contract of guaranteed service

In Chapter 3, we have collected evidence that strict guarantees as provided and required by guaranteed service will not be satisfactory in real networks. Guaranteed service requires that all network elements on the path export how their service deviates from the fluid model. The sum of the exported worst-case delays of the network elements and their adjoining links is assumed to be the worst-case delay for the entire path. There are three reasons why we believe that this estimated total worst-case delay is not well suited as a quantitative guarantee.

First, it is often impossible to estimate upper limits on delay in a network element. The worst case may or may not be known. For instance, the maximum waiting time for a slot on a time-sliced link can be easily calculated. Also, the maximum service interruption caused by a routing update will be known. In other cases, the worst case is caused by unexpected behavior or events, as observed with the router-induced jitter described in Section 3.2.3. Unexpected delays cannot be estimated, resulting in overly optimistic worst-case delays and possible violation of reservations.

Second, it is impossible to control link layer queuing or to estimate delay bounds for link layer elements, as discussed in Section 4.3. Link layer devices are transparent for network layer protocols, but queuing at the link level might lead to significant delay variation. On most legacy LANs, it is impossible to provide service guarantees.

Third, the total worst-case delay of the path is the sum of the individual worst-case delays. Although it is very unlikely that a packet experiences worst-case delay in all network elements, a guarantee must take this case into account. The total worst-case delay can easily add up to several seconds. A delay guarantee of several seconds will render the service useless. It is hard to imagine a "real-time" application that is willing to accept delays of several seconds but cannot tolerate any packet loss. A reliable TCP connection would be more appropriate for such an application.

For these reasons, we believe that users will not find guaranteed service particularly useful. Delay bounds are either not reliable, or they are too high for the service to be of any use.

Service providers also have good reasons to be hesitant about guaranteed service. First of all, it is fairly difficult to automatically discover and advertise the worst-case delay on any given path. Second, each guaranteed service flow needs to be completely isolated. The reserved bandwidth of a guaranteed service flow belongs entirely to that flow during overload. No delay is shared with other flows. Weighted fair queuing (WFQ) is a common scheduling algorithm to realize isolation. Each guaranteed service flow is serviced by its own WFQ queue. WFQ scheduling is a heavy burden on router performance and does not scale at all. Present-day routers cannot handle more than a few dozen fair queues on a fast interface (see Section 4.5). The required isolation also prohibits aggregation of guaranteed service flows, so scaling problems are difficult to solve. (Aggregation of guaranteed service flows was proposed only for the special case in which flows traverse exactly the same path and have very similar QoS requirements, see [Ram96].)

Another issue is that the niche for guaranteed service is very small. For mission-critical applications, dedicated lines and networks with backup mechanisms will remain the medium of choice. As we have seen, guaranteed service can't deliver 100% guarantees. Internet applications need to be adaptive even if they use guaranteed service. Even the most sophisticated scheduling mechanism cannot completely eliminate the effects of resource sharing. Thus, one could say that guaranteed service defines itself out of existence.

4.5 Scaling issues of Integrated Services/RSVP

There are at least three scaling issues involved in large scale deployment of RSVP: scaling of control traffic within a multicast group, scaling of reservation state for many reservations and scaling of reservation enforcement for many reservations. On the lowest level, the control traffic and reservation state within a single large multicast session should be limited. This problem was solved by the RSVP design. In multicast sessions, path messages are also sent as multicast messages, thus minimizing traffic. Reservation requests are merged at each branch point of the multicast distribution tree, so there is only one resv message on each branch of the tree (per refresh interval). Each additional reservation request in a large multicast group travels only a short distance before it merges with another reservation request. Shared and wildcard filters allow aggregation of reservations for flows with the same destination. Thus, the amount of control traffic and reservation state scales better than linearly with the number of receivers in a single multicast session. If wildcard filters are used, control traffic and reservation state also scale better than linearly with the number of senders in one multicast session.

[Mit95] further analyzes scaling aspects of RSVP within a single multicast session. The author concludes that RSVP's support for heterogeneous receiver requests and multiple reservation styles contributes to lower network resource requirements (when compared to ST-II, see Section 6.5).

The second scaling issue is managing the reservation state for a large number of sessions. The number of RSVP control messages processed by each router is proportional to the number of QoS flows going through the router. RSVP deals with application-level flows, such as one multicast audio session or a video transmission from a single source. Reservation state is kept on a per-flow basis. Thus, managing state and processing control messages scales linearly with the number of flows. However, managing reservation state puts a heavy strain on routers with large interfaces. Information about thousands of reservations needs to be stored, accessed and changed. The primary function of routers is packet forwarding. Managing state information and performing additional look-ups necessarily degrades router performance. The management capabilities of RSVP routers must scale in proportion to their forwarding path bandwidth to fully utilize the capacities. Unicast routing tables store information per destination, aggregated by hierarchical routing. Multicast routing tables store information per multicast session and possibly per sender, depending on the multicast routing protocol. Aggregation of multicast routing state across groups is impossible with the current addressing scheme. In addition to unicast and multicast routing state, RSVP-capable routers need per-flow state, further straining the router's management capabilities. The amount of unicast and multicast routing state in a router depends only on the network topology and is insensitive to the size of the router's links. In contrast, the state required for RSVP (reservation state and path state) grows with the bandwidth of the links. The larger the links, the more flows can be served, but the more state information needs to be managed. Compared to unicast routing state, reservation state is relatively short-lived and thus frequently changed.

With the deployment of fast routers with large memory, routers might be able to handle RSVP state management within domains in spite of the poor scaling properties of RSVP reservation state. As links become even larger and support even more reservations, however, it is unlikely that the management capabilities can keep up. It is not expected that large routers on inter-domain backbones keep per-flow state. Some form of aggregation will be necessary.

The third scaling problem is enforcement of reservations. All incoming packets go through a packet filter. The packet classifier has to check the list of reservations for each packet to determine which service the flow should get. According to the current filter specification ([BZB$^+$97]), a packet is classified according to fields in its network and transport layer headers. This is not only a violation of layering, it is also very expensive because the classifier has to look far into the packet. The reservation style determines which fields are relevant for classification. Thus, the cost of classifying is proportional to the number of packets going through the router, not to the number of reservations or the number of flows.

The cost of packet scheduling depends on the number of different services the router supports. The more queues and priority levels there are, the more expensive is packet scheduling. A packet scheduler basically performs a sorting operation on all incoming packets. On large interfaces, the bottleneck is usually not the bandwidth of the outgoing link, but the processing speed of the router. This is true even without sophisticated scheduling mechanisms. Present-day routers aren't able to handle more than a few dozen different queues with weighted fair queuing on a fast interface. For example, in one of our experiments a Cisco router of the 7500 series broke down when fair queuing with 128 queues was turned on on an FDDI interface.

Even now, without reservations, routers are considered to be bottlenecks in internets. Router vendors use caching of frequently used data to improve performance. Some critical functions are implemented in hardware or done off-board. One problem with Integrated Services is that they are too flexible to lend themselves well to optimizations. Each flow requests its own quality of service and gets special treatment from the router. Integrated Services allow QoS parameters to be chosen from a continuous scale, thus support an "infinite" number of different QoS levels. Dynamically changing numbers of queues and QoS classes make optimizations difficult to implement.

CPU-intensive queuing strategies together with large interfaces are an architectural no-win situation. With increasing bandwidth the number of flows and requested services grows. Not only does the router have to forward more packets per second, but the forwarding decision is also more difficult because of a larger number of queues and QoS classes.

The next chapter presents several approaches to lighten the load on RSVP routers and to make them faster.

Chapter 5

Approaches to solve the scaling problems of Integrated Services/RSVP

This chapter presents aggregation and switching as two approaches to solve the scalability problems of Integrated Services/RSVP. We develop a new form of aggregation and examine the applicability of two commercial switching approaches that were designed with goals other than RSVP in mind.

5.1 Aggregation

Managing per-flow reservation state in large backbone routers is very expensive. For every packet, the classifier module has to look up the reservation information for the flow. Aggregation is needed to reduce the amount of reservation state and the cost of classifying. Aggregation means treating several RSVP flows as one. Of course, only flows with similar QoS requirements can be aggregated. Flows that are aggregated into a superflow share the delay. Isolation is not possible for aggregated flows. Aggregation can be done in various degrees. One form of aggregation was proposed in [Boy97] for cases where several sending members of the same multicast group are located at the same site. Path messages and reservation requests can then be aggregated using the common classless inter-domain routing (CIDR) prefix. Data packets are filtered according to their CIDR prefix, not according to their full-length sender addresses. The

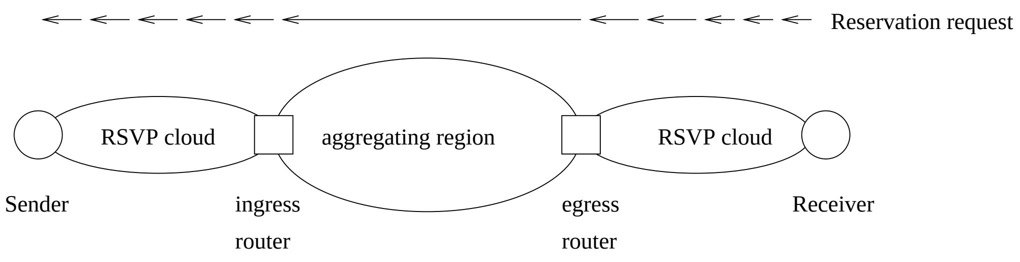

Figure 5.1: "Classy" aggregation

most common application with these characteristics is distributed simulation. CIDR aggregation is also potentially useful for virtual private networks (VPNs).

Another internet draft ([Ram96]) proposes an aggregation scheme for guaranteed service flows that use exactly the same path and have very similar QoS requirement. Both schemes offer improvements in special cases, but what is really needed is a general solution for over-strained backbone routers.

5.1.1 "Classy" aggregation

In [BV97], Berson and Vincent propose a "classy" approach to aggregation. *"Classy" aggregation* is a hybrid between native RSVP in the networks near the clients and aggregation in the backbone. *Aggregating regions* (e.g. a routing domain within a provider network, one or more contiguous autonomous systems) offer a small fixed number of service classes. On entry into the aggregating region, each flow for which a reservation was made is assigned to one of the service classes. Flows with similar service requirements are grouped together in a service class. Service class definition and flow assignment is the subject of ongoing research. Each packet is marked with a tag that identifies which service the flow should receive. For IP, this tag could consist of the Type of Service (TOS) bits in the packet header or the packet could be encapsulated. Inside the aggregating regions, packets are scheduled according to their assigned service class. Because the number of classes is fixed, packet scheduling is less expensive. Classification is done according to the tags, minimizing classification state. No processing of RSVP messages is necessary in the interior of aggregating regions, eliminating reservation state. In this latter respect, aggregation regions are similar to transparent non-RSVP clouds. There is no admission control for non-RSVP clouds, however, while aggregating regions do admission control and policy control for the entire region as a whole. When an RSVP

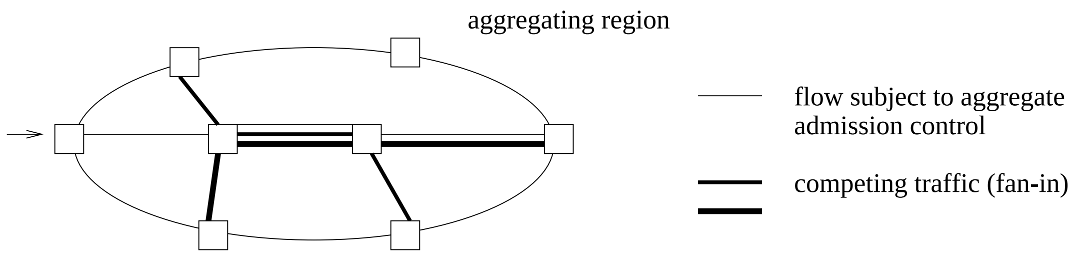
aggregating region
flow subject to aggregate admission control
competing traffic (fan-in)

Figure 5.2: Fan-in from other ingress routers can congest downstream links

reservation request arrives at an ingress of the aggregating region (the edge router closer to the sender of the data, see Figure 5.1), and that reservation passes policy control and admission control for the entire region, then data packets from that flow are assigned to a traffic class and marked with the appropriate service class tag. .

If packet scheduling is implemented accordingly, the region isolates different service classes from each other. But admission control is needed to avoid congestion of links and within classes. It is an open question how admission control for the entire region can be done. The ingress router can only control the load on its own interfaces. Traffic from other sources that enters the aggregating region through different ingress routers might congest downstream links, as shown in Figure 5.2.

Aggregate admission control could be done in several ways, involving different trade-offs between wasted bandwidth (i.e. bandwidth that can only be used for best-effort traffic) and the risk of overload.

A simple method is to configure certain maximum bandwidth fractions for each service class into each ingress router. For example, each of four service classes could get a maximum of 20% of the link bandwidth, leaving 20% for best-effort traffic. Reservation requests are rejected when the class is full at the ingress router. If these maximum bandwidth fractions are chosen such that congestion in the aggregating region is completely prevented, a lot of bandwidth is wasted. Otherwise, this scheme cannot prevent overload of classes on downstream links. Therefore, flows for which reservations have been made might not get their reserved QoS. Still, admission control reduces the probability of congestion.

If the aggregating region consists of a frame relay or ATM network, the problem of aggregate admission control is shifted to a lower layer. According to their reservation, RSVP flows use one of several virtual circuits. The ATM generic admission control or frame relay traffic control is responsible for preventing overload in the interior of the network. RSVP controls the QoS parameters to set up virtual circuits. However, ATM traffic contract enforcement does not solve the problem of aggregate admission control if flows with different service requirements share the same SVC. An ATM interface simply drops all traffic that exceeds the capacity of the virtual circuit, without regard to layer 3 QoS requirements. Therefore, the border router must ensure fair use of the SVC according to the reservations. This could be done by transforming the Tspecs of the reserved flows not only into appropriate ATM QoS parameters, but also into priority levels. The border router then enforces these priorities with class-based queuing (CBQ) and weighted RED, so that packets of misbehaving sources are progressively dropped. The CBQ and weighted RED mechanisms are explained in more detail in Section 6.3.

In a special case, there would be only two RSVP routers on the path: the first-hop router and the last-hop router, connected by a virtual circuit. [Nol96] calls this setup *virtual circuit meshing.* Strictly speaking, the entire frame relay or ATM network really is only one hop, not a region. As ATM is becoming more widely used for backbones in intranets, [Nol96] views virtual circuit meshing as a promising approach to combine ATM and RSVP while avoiding the performance penalties of intermediate routers. Heterogeneity issues concerning RSVP over ATM will be discussed in Section 5.2.1.

In summary, "classy" aggregation eliminates reservation state in aggregating regions and simplifies packet classifying and scheduling. It significantly improves the scalability of RSVP. However, aggregate admission control either wastes bandwidth or risks congestion so that no-loss guarantees might be violated.

5.1.2 "Classy" aggregation with full reservation state

The performance of routers grows steadily. Routers are now able to manage huge amounts of routing state. For example, multicast routing protocols require at least one routing entry for every active multicast session. Some routing protocols even require an entry for each sender in every multicast session ([MS97]). Of course, this does not scale Internet-wide, but the MBone has reached a remarkable size in spite of poorly scaling multicast routing protocols. Under the assumption that only a few receivers in a few percent of all multicast groups will request a reservation, the amount of reservation state that routers have to manage will be less than the amount of multicast routing state. Managing reservation state in routers is costly because the information has to be accessed for every packet with or without a reservation. The main problems are not memory requirements or update rates. The primary problem is the high access rate. The following approach reduces the access rate but maintains full reservation state which is necessary for accurate admission control.

I propose *"classy" aggregation with full reservation state* to reduce the cost of packet classifying and scheduling while maintaining RSVP's accurate admission control and bandwidth assurances. The aggregate admission control of “classy” aggregation either wastes bandwidth or risks congestion in the aggregating region. The modified approach does not eliminate reservation state. Reservations are set up hop-by-hop as usual, even in the aggregating region. Routers maintain reservation state for every flow. Thus, admission control is accurate and reliable. As with “classy” aggregation, each flow is assigned to one of a few service classes. When processing the reservation request, each RSVP node subtracts the accepted reservation from the available bandwidth in the assigned class. Rejected reservation requests trigger error notifications as usual. If a reservation was accepted, the client can be sure that the reserved flow will get the requested bandwidth and quality of service. This is the main advantage of this proposal over Berson and Vincent's “classy” aggregation. As with “classy” aggregation, data packets are tagged at the ingress router to indicate their service class. In the interior of the aggregating region, data packets are classified and scheduled according to their service tag, so there is no need to look up reservation state for every data packet. Because of the fixed number of service classes, the cost of classifying and scheduling is small and independent of the number of flows. The reservation state is only accessed when the node receives another reservation message. The RSVP demon determines whether it is just a refresh message, or if it is a reservation request that can be merged with an existing reservation, of if it is a new reservation. Reservation state times out as usual.

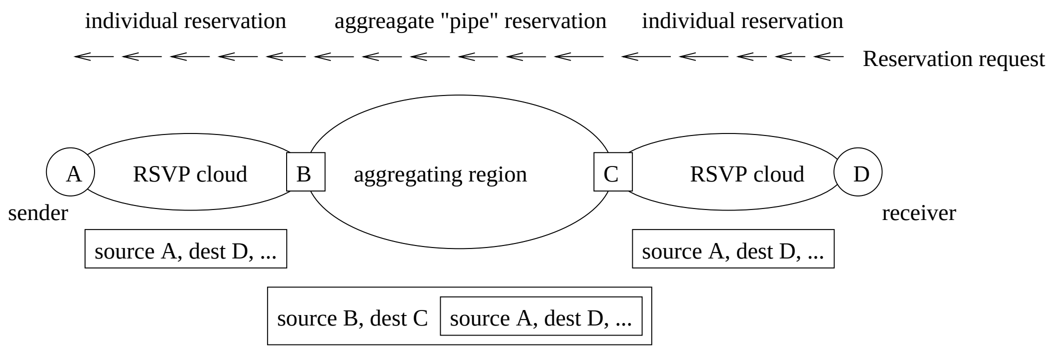

Figure 5.3: Hierarchical RSVP

Heterogeneous reservations in aggregating regions are an open issue in both the modified and the unmodified version of “classy” aggregation.

In summary, “classy” aggregation with full reservation state combines the advantages of “classy” aggregation and RSVP’s admission control. It significantly reduces packet classifying and scheduling costs. Overload of service classes is prevented by maintaining hop-by-hop admission control. Therefore, RSVP clients will in fact get the service they expect.

5.1.3 Hierarchical RSVP

Hierarchical RSVP is an idea that is mentioned from time to time, but no proposal has been published yet. While set-up and tear-down patterns of single RSVP flows are unpredictable, the aggregation of hundreds of flows seems more stable. The idea of hierarchical RSVP is that routers at the edge of aggregating regions use RSVP to reserve large “pipes” in a few QoS choices through the region. At the ingress router (see Figure 5.1 again for an illustration of ingress and egress), data packets are assigned to a pipe according to their service requirements and destination. The packets are encapsulated so they can be classified and scheduled as part of the pipe. Source and destination of the encapsulated packets are ingress and egress routers. Figure 5.3 illustrates this concept. Again, there are only a limited number of different qualities of service available. Since RSVP is receiver-oriented, pipe reservations have to be made by egress routers. Egress routers could reserve a number of pipes at system start-up and then adjust the reservations as the actual demand becomes known. When the demand changes, pipe reservations can be further adjusted. A pipe reservation is only maintained if there is a sufficient number of reserved flows that use the pipe. Therefore, a router doesn't necessarily have to have a pipe to every other router, leading to better scaling of the mechanism. Reserved flows for which no pipe exists are served as usual, without aggregation.

The advantage of hierarchical RSVP as described here is the reduction of reservation state. Routers in the interior of aggregating regions only keep reservation state for the larger pipe reservations. Packet scheduling is simplified by offering only a few QoS choices. The main disadvantage is that packet classifying is still done by looking far into the packet headers and comparing source and destination against the (now shorter) list of reservations. Also, the pipes are point-to-point, so heterogeneity in the interior of the aggregating region cannot be supported.

Recently, a variation of this idea was presented at an RSVP working group meeting under the name "RSVP tunneling" ([KWTL97]).

5.2 Switching

Routers are becoming bottlenecks in networks. They can't keep up with the increased traffic capacity of high-speed switches. Added intelligence in routers further slows down packet forwarding. Conventional routers compute data paths (and scheduling information, if the router supports differentiated service) on a packet-per-packet basis. Internetworking vendors have announced several new routing solutions that combine network layer routing with link layer switching. Decisions are then made per flow, not per packet, and most network layer processing can be eliminated. Promising approaches are Ipsilon Networks' IP Switching as well as Cisco Systems' Tag Switching and NetFlow Switching.

5.2.1 IP switching

IP switching is a proprietary but published and open technology developed by Ipsilon Networks ([NEH$^+$96], [IP 96]). It is designed for ATM-based IP networks. Support for frame relay is planned. IP switching aims to optimize IP throughput by switching most traffic across the ATM network, bypassing the routing infrastructure.

IP switching works as follows. Each IP node sets up a *default virtual channel* on each of its ATM physical links. This virtual channel (VC) is used to forward packets in the normal, non-optimized manner. The first packets of a flow are always forwarded on the default VC. An *IP Switch Controller* decides which of the packets arriving on the default VC belong to a long-lived flow, like ftp, telnet, WWW or real-time data. (Note that the IP Switch Controller considers a flow of several seconds long-lived.) Long-lived flows are worth optimizing by giving them their own virtual channel and switching them on the ATM level. Short-lived traffic (e.g. DNS queries, SNMP queries, SMTP data) continues to be forwarded on the default VC. A flow is characterized by a source - destination pair and other header fields as configured. Once the IP Switch Controller has identified a flow, it asks the upstream IP node to send that traffic on a new ATM virtual channel. Independently, the downstream IP Switch Controller will have identified the flow in the same way and requests that the traffic be sent on a new virtual channel. At this point, the flow does not use the default VC any more. It is isolated to a particular input channel and a particular output channel. The flow can then be optimized by *"cut-through" switching* in the ATM hardware, bypassing the routing software and the associated processing overhead. When a flow is switched on the ATM level, packets do not need to be reassembled from cells. This decreases transmit delays. The general concept is that long-lived flows are switched on the ATM level while short-lived traffic is routed as usual. The efficiency of IP switching depends on the traffic pattern. IP switching works best when a high percentage of traffic can be classified as long-lived flows. The longer the duration of the flows, the less overhead there is for setting up and tearing down virtual channels.

IP switching and RSVP

Ipsilon Networks announced that RSVP support was planned for the future. A straightforward approach is to make all IP switches RSVP-capable. A reservation would cause a VC with the desired QoS to be set up. If the reserving flow is already served by its own VC, it is shifted to the new QoS VC. All IP switches need to have an RSVP demon and admission control. All flows for which reservations are made meet the definition of a long-lived flow, thus they get their own VC and are switched on the ATM level. Therefore, layer 3 classifier and packet scheduler modules are not needed in IP switches. However, all general issues of RSVP over ATM (see [BB97]) also have to be addressed.

The limitations in ATM's support for multicast are one of the problems of RSVP over ATM. In ATM's point-to-multipoint connections, all receivers get the same QoS. Heterogeneous reservations have to be accommodated by providing the largest reservation to all receivers or by duplicating traffic on different channels. If there are receivers with reservations as well as receivers with best-effort service, two policies are possible. Best-effort traffic is accommodated either by separate VCs for best-effort and QoS traffic, leading to two copies of a packet on the same link, or by giving it a sometimes free, sometimes expensive ride on the reserved QoS VC.

These heterogeneity issues can cause problems on large IP switched networks. IP switching is designed to take advantage of the benefits of larger switched ATM networks while avoiding the disadvantages of a large flat network. The main disadvantage of a large flat ATM network is that all routers at the edge of the ATM network are each others' routing peers, causing scalability problems with the number of routing peers. ATM switches do not participate in routing protocols, but IP switches do, so the number of routing peers is smaller and thus scales in an IP switched ATM network. In effect, an IP switched network looks flat to long-lived flows and hierarchical to the routing peers. Thus, IP switching allows to build larger ATM networks. However, the RSVP heterogeneity issues become more serious with the size of the IP switched ATM network. In larger networks, it is more likely that several receivers request different QoS. Subpaths that carry duplicate traffic are longer if the approach of separate VCs for different QoS is chosen. If heterogeneous QoS requests are handled by giving the better QoS to all receivers, more resources need to be used that were not requested. It is also more likely that a larger reservation for a new receiver is not accepted although there are enough resources available between the source and the VC end-point of this new receiver. This occurs when any of the other existing QoS VC endpoints with smaller reservations can't upgrade to the new large QoS.

Figure 5.4 illustrates an example situation in the *limited heterogeneity model*, where receivers of a multicast session are limited to use either best-effort service or a single alternate quality of service. In Figure 5.4, sender S sends to a multicast group that was joined by R1, R2, R3 and R4. Receivers R1 and R4 have a reservation, while receivers R2 and R3 get best-effort service. Between router r0 and switch s2, two copies of each packet are sent, one on the best-effort VC, one on the QoS VC. Let's assume that the link between switches s1 and s2 works at almost full capacity because of some other

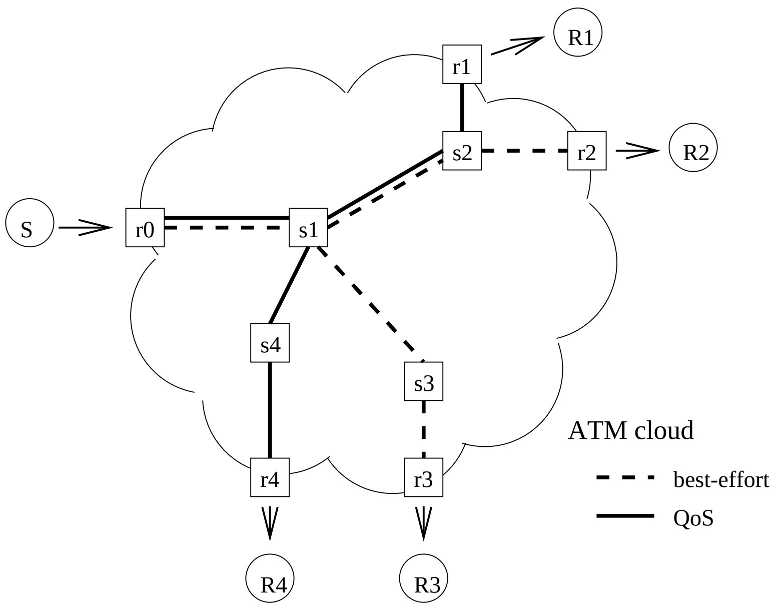

Figure 5.4: Heterogeneity issues for RSVP over ATM

traffic. Now receiver R3 requests a QoS that is larger than the existing QoS VC. This requires that a new point-to-multipoint VC with larger QoS is set up to serve R1, R2 and R4. Resources for that QoS are available on the path between routers r0 and r3, but not between switches s1 and s2. Thus, R3's reservation request fails, although there are enough resources available between the sender and R3.

Scalability

IP switching with full RSVP support solves the scaling problem concerning reservation enforcement. Once a VC with the desired QoS is set up and the flow is serviced by cut-through switching, packet classification and scheduling are eliminated. Nodes still have to manage all reservation state, however.

Another remaining scaling problem is independent from RSVP. A virtual channel is set up for each flow. Depending on the traffic pattern, this can be too much strain on backbone IP switches that handle thousands of flows. RSVP causes even more VC set-ups, because flows are shifted from their original VC to a QoS VC to honor the reservation. On the other hand, RSVP flows are likely to be long-lived. The scaling problem is caused mainly by flows with a duration of only several seconds, for example file transfer flows. To solve this scaling problem in backbones, some form of aggregation will be necessary. But because IP switching relies heavily on the concept of a flow, there seems to be no obvious way to aggregate.

Static QoS

As an alternative to RSVP, today's IP switches offer the possibility to provide QoS through static configuration of the switches. When the IP Switch Controller identifies a flow, it assigns a priority or one of a few QoS levels to the flow. The configuration of the switch determines which packet header fields are relevant for assigning a QoS level. Then, a VC with the assigned QoS level is set up for the duration of the flow. ATM ensures that the QoS of a VC remains about the same during the life-time of the VC. When the entire capacity of an ATM link is allocated to VCs, subsequent set-up requests are rejected. The disadvantage of providing static QoS is that it is inflexible. Quality of service is assigned according to static information that has to be configured into the switches. QoS cannot be requested by applications.

IP switching, "classy" aggregation and static QoS combined

An IP switching network with different QoS levels has the potential to work well as an aggregating region in Berson and Vincent's "classy" aggregation scheme. No RSVP messages are processed in the interior of the IP switched network. On entry into the aggregating region, data packets are tagged with a QoS identifier according to the reservation. Once the IP switched network has identified the flow, it sets up a virtual channel of that quality between ingress and egress router.

Admission control is difficult to do at the ingress router, however. A VC of the appropriate quality is only set up after the IP switches have seen a sufficient number of packets. If the VC set-up fails, an error message has to be sent back to the ingress router and from there an error notification is sent to the requester of the reservation. Consequently, reservations can fail shortly after they have been set up. But if they don't fail, the QoS will remain stable over the lifetime of the flow due to the generic admission control of ATM. Compared to IP switching with full RSVP support, this scheme eliminates reservation state in IP switched networks. However, the high rate of

VC set-up and tear-down remains a severe scaling problem.

5.2.2 Tag switching

Tag switching is a proprietary technology proposed by Cisco Systems ([RDK+97]). Among other goals, it aims to simplify the forwarding decision of routers by using a label-swapping technique, thus improving performance. Tag switching can be implemented over many media types and is independent of network layer protocols.

Tag switching works as follows. At the edge of a tag-switched network, a tag is applied to each packet. A tag has only local significance. When a packet is received by a *tag switch* (a router or ATM switch with tag switching software), the switch performs a table look-up in the tag table (*Tag Information Base*, TIB). An entry in the tag table consists of an incoming tag and one or more tuples of the form (outgoing tag, outgoing interface, outgoing link level information (e.g. MAC address)). The tag switch replaces the tag in the packet with the outgoing tag and replaces the link level information. The packet is then sent out on the given outgoing interface. The Tag Information Base is built at the same time as routing tables are populated, not when the tag is needed for the first time. This allows flows to be switched starting with the first packet. Label swapping is much faster than routing because the network layer is not involved. Switching bypasses the router's processor. Label swapping can be done in constant time, because it is exact match. In contrast, a routing table lookup is best match and thus in the order of $O(\log n)$.

A control component is responsible for binding tags to routes. Depending on the desired granularity, a tag can be associated with a single flow, with a single route or with a group of routes. All tag switches fully participate in routing protocols. A tag switch builds its Tag Information Base by associating tags with routes from the routing table (*Forwarding Information Base*). One method to allocate and distribute tags for destination-based routing is called *downstream tag allocation*. Figure 5.5 illustrates the process. A downstream tag switch allocates an incoming tag for each route in its routing table. It then advertises the (incoming tag, route) - pairs to its peers. When a tag switch receives tag binding information for a route from the next hop on that route, it accepts the incoming tag of the downstream switch as its own outgoing tag for that route. This creates the binding between incoming tag, route and outgoing tag. Tag

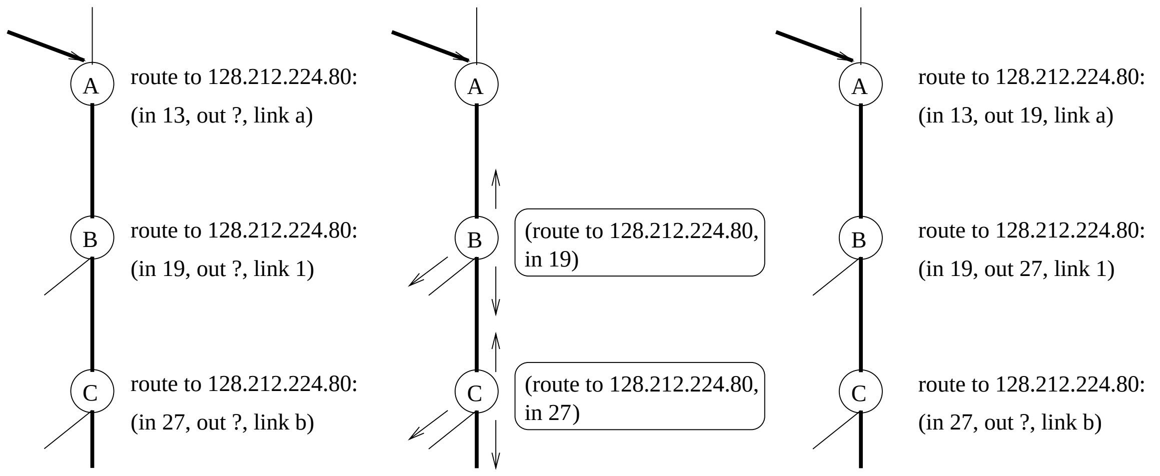

Figure 5.5: a) Switches allocate incoming tags, b) B and C advertise tag information, c) The upstream hops for the route place the tag in TIB

binding information is either distributed through a separate *Tag Distribution Protocol* (TDP) or by piggy-backing advertisements on top of existing routing protocols.

Tag switching and RSVP

Tag switching allows to adjust the granularity of tag allocation. For simple destination-based routing, a tag is assigned to each destination prefix. To ensure per-flow QoS, a tag is assigned to each session for which a reservation is made. Tags can be distributed as tag objects in RSVP reservation requests, similar to the downstream tag allocation scheme described before. This requires an extension to RSVP. Upon receipt of a reservation request message, a tag capable router assigns a tag to the flow, then passes the tag upstream with the reservation message. A tag for an RSVP flow would include the QoS for that flow. The tag would be used both to make a forwarding decision and to make a scheduling decision, e.g. by selecting the appropriate queue.

To ensure proper tag allocation, all routers/tag switches on the path must be RSVP capable. The property of transparent non-RSVP clouds is lost. At a non-RSVP capable tag switch, the chain of labels assigned to a reserved flow breaks and the tag switch has to apply a tag using destination-based routing. This causes delay, because the packet is routed, not switched. Moreover, once packets follow the best-effort chain of tags, tag switches "blindly" switch them to their destination as best-effort traffic, even though downstream tag switches might be RSVP-capable and have allocated tags for that flow.

For that reason, either all or none of the routers in a tag-switched network should be RSVP capable.

Another problem is setting up path state. Path messages are always sent to the same destination address as the data. Before a reservation is set up, path messages are switched on the best-effort chain of tags according to destination-based routing. Because the goal of tag switching is to bypass network layer routing, tag switches just look at the tags, not at packet headers. Unless otherwise told, tag switches don't know whether a packet is a data packet or an RSVP path message that should cause them to set up path state. To solve this problem, the tag edge router must assign a special tag to path messages. One approach would be to use a special tag that means "Do not tag-switch", so routers are forced to look into the packets and can take the appropriate action.

When tag switching is implemented over ATM, VC identifiers are used as tags. Tag switching is then done on the ATM level. All heterogeneity issues described for IP switching also apply to tag switching. Most other media allow different QoS on each branch of a multicast tree, thus heterogeneous reservations don't present a problem.

Scalability

Tag switching reduces the complex task of packet classifying to one look-up in the Tag Information Base. For that reason, tag switching solves RSVP's scalability problem concerning packet classifying. However, other scaling problems remain. Although the tag determines the scheduling decision, the scheduling itself is as expensive as without tag switching (unless the scheduling is done on layer 2, as with ATM). A varying number of queues with various queuing strategies must be administrated and serviced. The cost of scheduling grows with the number of different services that are offered. Unless some form of aggregation is used, tag switches also have to manage the reservation state on a per-flow basis.

Tag switching provides for aggregation by allowing hierarchical tags. A packet can carry not one, but a stack of tags. This feature was designed to support hierarchical routing protocols. It could also be used to support aggregation of RSVP flows. The tag at the bottom of the stack would be a per-flow tag. In aggregating regions, packets that get the same service and go through the same egress router of the region get the same tag. When leaving the aggregating region, flows are demultiplexed by popping the aggregate tag from the stack. As with hierarchical RSVP, aggregate admission control and reservation set-up are open issues.

Static QoS

Tag switching can also be used to statically support a small number of classes of service. For example, a tag switch could allocate not one, but two tags per destination, one for standard/best-effort service, one for premium service. A configured tag edge router would classify packets in one of these two classes according to some fields in the header. Again, as with IP switching and static QoS, this scheme is inflexible and has nothing to do with the “QoS on demand” approach of RSVP.

5.2.3 NetFlow switching

Cisco Systems developed *NetFlow switching* ([Net96]) as another technique to reduce per-packet overhead associated with routing. The idea of NetFlow switching is to identify end-to-end flows and then apply these services to the flow, not to single packets. NetFlow switching improves local router performance. No communication with other routers is necessary. Access control and other security tasks, accounting and sophisticated queuing strategies all reduce performance. To speed up routing of RSVP flows, packet scheduling on a per-flow basis would be most useful. Cisco plans to offer this in the future by integrating NetFlow switching and weighted fair queuing. NetFlow switching could also be a first step towards per-flow accounting.

5.3 Non-RSVP clouds with static QoS

The simplest method to eliminate scaling problems is to treat a large backbone as a non-RSVP cloud. RSVP is designed so that non-RSVP clouds are transparent and can be “tunneled” without encapsulation. Within a non-RSVP cloud, all traffic gets best-effort service. This approach is taken today. However, congestion mainly occurs on the backbones, so reservations have little effect if the backbone is a non-RSVP cloud. On backbones that offer different levels of QoS, the situation can be improved by assigning a reserved flow to an appropriate QoS level. Some networks use tags or Type of Service bits to distinguish between QoS levels. There is no mechanism to prevent overload.

Consequently, no guarantees can be given regarding the end-to-end QoS. However, chances are that a reserved flow receives a service that is better than best-effort service. Most new routing solutions (e.g. IP switching, tag switching) provide the capability to differentiate between a few different qualities of service.

In cases in which the RSVP-capable parts of the network are prone to delay variations and packet losses, this combination between RSVP and static QoS makes sense. Partial control is better than no control. On the other hand, RSVP loses much of its appeal when it doesn't provide predictable end-to-end QoS. In cases in which the main QoS problems originate in the backbone, the overhead of supporting RSVP will not be worthwhile. Chapter 6 discusses priorities and class of service approaches as alternatives to RSVP.

5.4 Summary of approaches

	Advantages	**Disadvantages**
1. CIDR prefix aggregation, [Boy97]	• reduces number of reservations	• not a general approach, only for niche applications
2. Aggregation of guaranteed service flows, [Ram96]		• not a general approach
3. "Classy" aggregation, [BV97]	• bounded amount of classification state • less expensive scheduling • no reservation state • no RSVP message processing	• aggregate admission control either wastes bandwidth or risks congestion • heterogeneity issues unsolved
4. "Classy" aggregation with full reservation state	• bounded amount of classification state • less expensive scheduling • fine grain admission $\rightarrow$ predictable end-to-end QoS	• part of the scaling problem of reservation state remains • heterogeneity issues unsolved
5. Hierarchical RSVP	• reduced reservation state • simplified packet scheduling	• complex flow assignment and pipe management • heterogeneity in aggregating region not supported • packet classifying still expensive
6. IP switching with full RSVP support	• no scheduling and classifying on layer 3	• setting up a VC for each flow does not scale • part of the scaling problem of reservation state remains • heterogeneity wastes bandwidth or leads to undesired behavior

	Advantages	**Disadvantages**
7. IP switching, "classy" aggregation and static QoS combined	• combined advantages of 3. and 6. • stable QoS	• setting up a VC for each flow does not scale • reservations can fail shortly after set-up
8. Tag switching with full RSVP support	• packet classifying is easy	• packet scheduling is expensive • full reservation state has to be managed
9. Non-RSVP clouds with static QoS	• simple • all scaling problems eliminated	• no predictable end-to-end QoS, no guarantees • service might or might not be better than best-effort service

Chapter 6

Alternatives to RSVP and Integrated Services

Let us take a step back and look at the original problem that RSVP was designed to solve. In present-day networks, congestion often leads to unsatisfactory quality of real-time transmissions. The fundamental goal is to provide good quality of service for real-time flows in packet-switched networks.

As Figure 6.1 shows, this goal could be achieved with various strategies. One could aim to avoid congestion for all traffic classes or just for real-time traffic. The simplest method to prevent all congestion is *over-provisioning*, i.e. providing more bandwidth than will ever be needed. This is very expensive, and experience has shown that it takes only about two years before utilization of resources climbs close to 100% again. Another solution might be to let the laws of a free market economy take care of the bandwidth problem. Users and service providers would be billed for the amount of traffic they inject into a provider's network. *Volume-based billing* is likely to reduce demand. It would then pay off for providers to expand their capacities. However, metered service does not seem to be popular in the Internet culture and is thus not expected to gain widespread acceptance until several years from now.

A very different strategy is to differentiate between real-time traffic and other traffic classes. This could be done with simple priorities, a fixed number of QoS classes or by dividing the resource pie up in advance. Resources can be allocated statically to a traffic class, a source/destination pair or an institution, as done in virtual private networks or CBQ-based resource allocation. Resources can also be allocated dynamically. This is

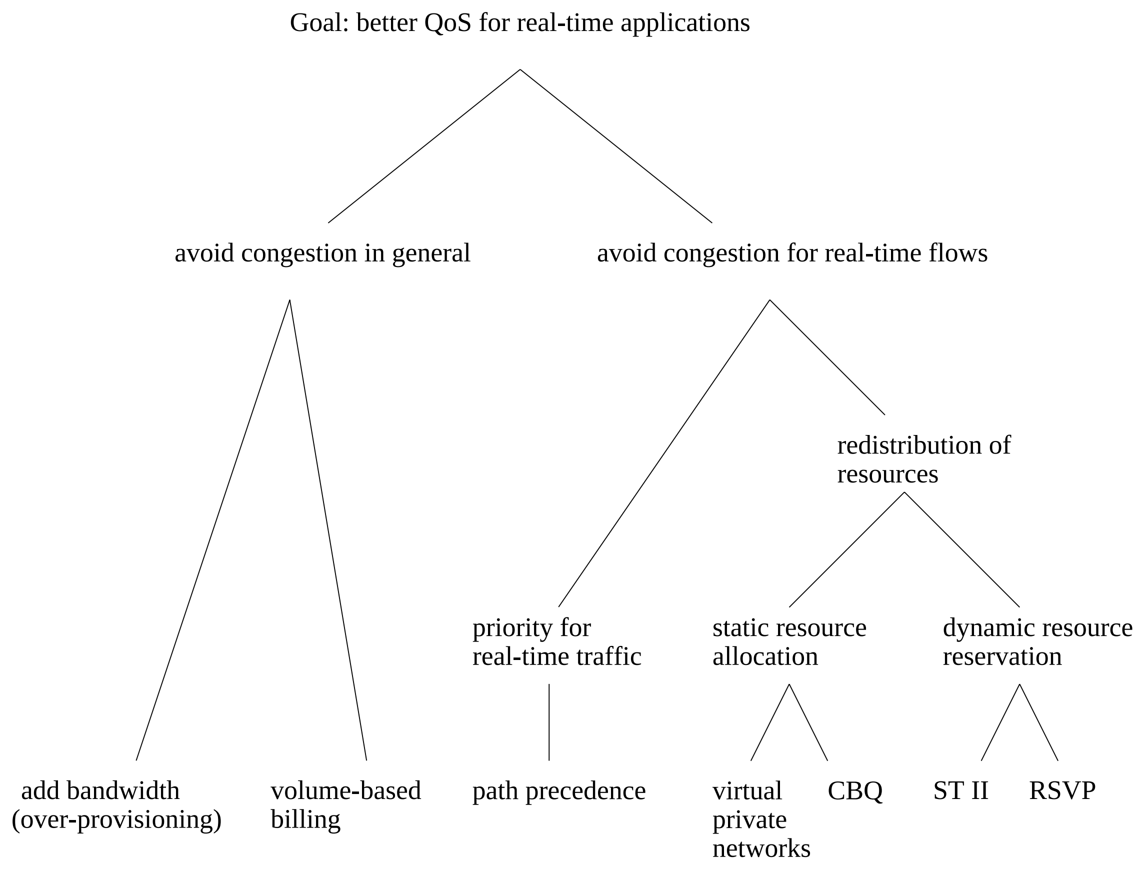
Goal: better QoS for real-time applications
avoid congestion in general
avoid congestion for real-time flows
redistribution of resources
priority for real-time traffic
static resource allocation
dynamic resource reservation
add bandwidth (over-provisioning)
volume-based billing
path precedence
virtual private networks
CBQ
ST II
RSVP

Figure 6.1: Strategies to provide better QoS for real-time applications

the approach taken by RSVP and ST-II. The following sections discuss these approaches in more depth.

6.1 Adding bandwidth and over-provisioning

Packet losses and delay variations are believed to be caused mainly by contention for network resources. Thus, an obvious solution is to provide sufficient resources so there is no contention. However, this is difficult because of the statistical nature of Internet traffic. Enough resources might be available on average, but when too many bursts coincide, network elements must queue or even drop packets. To ensure that this happens only very rarely, the network must be over-provisioned by a multiple of the bandwidth used on average, which is very expensive. On the other hand, if real-time applications are assumed to be tolerant and adaptive to a certain degree, contention does not need to be entirely prevented and over-provisioning for those requirements becomes more realistic.

One advantage of avoiding congestion by adding bandwidth is that it can be done locally. Only those network elements that prove to be congested frequently need to be upgraded. This approach is more selective and efficient than mechanisms that lead to overhead on every network element on the path, like RSVP.

Another important advantage of adding bandwidth is that it keeps all options open for subsequent technologies. All other strategies pose constraints on future mechanisms by requiring them to be compatible. For example, if a network supports priorities, all network devices deployed in the future also have to support priorities. If a network supports RSVP, all subsequent network protocols (e.g.IPv6) must be able to work with RSVP, at least conceptually. RSVP and other mechanisms will have to be changed for RSVP, in addition to the changes made in ordinary routers. Also, most other approaches do not scale easily to higher bandwidths. Even volume-based billing could pose a problem in that respect. It might be feasible to keep track of traffic origin and number of bytes sent on a multi-megabit interface, but not on a gigabit interface. Packet scheduling also suffers from scaling problems. Once a complex switching algorithm is in place on a router, it is very hard to add bandwidth. On a larger interface, the switching algorithm has to handle packets faster. In the case of Integrated Services, larger interfaces can accommodate more flows and thus usually have to support more queues. The list of reservations is longer, leading to more expensive packet filtering. Therefore, the switching algorithm does not only need to be faster, switching also becomes harder. Switching costs increase faster than he transmission speed. This in turn raises the cost of adding bandwidth. A high-speed router that can deal with the scheduling overhead of Integrated Services is likely to be much more expensive than a router that only supports simpler scheduling mechanisms.

Another way to add bandwidth is to build additional paths. However, redundant paths cannot be fully used with existing routing protocols. To locate alternate routes with sufficient capacity, a QoS-based routing protocol is needed. Finding an appropriate route will be a significant overhead if the total bandwidth is distributed over many alternate paths. It is more desirable to add bandwidth on existing links than to create too much redundancy, because redundancy leads to routing overhead and increases the size of the QoS-based routing tables. Moreover, the research community still discusses whether the benefits of QoS-based routing are significant enough to justify the complexity and cost of such a routing protocol.

The main disadvantage of over-provisioning is the high initial cost. In some cases, it might not even be possible to add bandwidth. In such situations, the existing resources have to be redistributed. "Throwing bandwidth at the problem" will improve real-time quality in some cases under certain circumstances, but the quality of service will not be predictable. Even if sufficient average bandwidth is available, jitter can significantly reduce the quality of a transmission. It depends on the characteristics of the desired applications whether a better-on-average service is sufficient or if precise predictions are needed.

In cases in which adding bandwidth leads to a lasting quality improvement for real-time applications, moderate over-provisioning might prove to be a cost-effective strategy in the long run.

6.2 Volume-based billing

In the Internet, charges have always been based on the size of the pipe leased by the provider or subscriber. Thus, there is no incentive not to fill up this pipe. Volume-based billing provides this incentive and has the potential to reduce demand. To smooth demand, fees could be structured according to the time of day. If the amount of traffic handled by a provider is directly connected to revenue, it will pay off for providers to expand their capacities. However, the Internet is not in a free market situation yet. It is considered unlikely that providers will start to impose metered usage charges, because they would likely lose customers to providers who don't. It is also not clear whether additional bandwidth will be available to upgrade links, since there seems to be a bandwidth shortage for certain kinds of links. Some people claim that the bandwidth shortage is artificial. The big players in the telecommunications industry are said to hoard unused raw bandwidth, because by selling it they would destroy their profitable telephone business. Other analysts believe this is no longer the case. They predict a "fiber crunch" in the near future. Already, it is difficult to purchase OC-3 connections (155 Mbit/s) consistently throughout the United States. Transatlantic and transpacific bandwidth is also scarce. No one really knows the total amount of available fiber, which makes it difficult to base investment decisions on hard facts.

We believe that bandwidth and other resources will always be scarce and need to be distributed in a more sophisticated manner than they were so far. If the bandwidth problem can be solved, most adaptive applications will get a satisfactory quality of service, although sufficient average bandwidth is no guarantee for good end-to-end quality.

6.3 Class of service approaches with precedence classes

In a fully connectionless model, each packet carries within itself the information needed to schedule it. In this sense, Integrated Services are not connectionless. State is required in routers to look up which kind of service a packet with a certain address and port receives. *Precedence classes* provide differentiated service without state in intermediate hops. Each packet carries an identifier that denotes the quality of service the packet should get. Routers differentiate between service classes according to a precedence field in the packet header. This is also called the *QoS/CoS* approach, where CoS stands for Class of Service. The IP protocol specification (IPv4) provides for a three-bit precedence field in the IP header called the Type of Service (TOS) field. In the "operational" Internet, this field has been rarely used, however, because the mechanisms to treat traffic preferentially were lacking. With the implementation of more sophisticated queuing strategies in routers, this situation has changed. [HR96] proposes to use the TOS field to differentiate between up to eight precedence levels, where zero is regular best-effort traffic and seven is "premium" traffic with the highest precedence. The exact meaning of "level n precedence" service depends on the scheduling mechanism. There are a number of different scheduling algorithms that all give better service to higher-precedence traffic.

One possible behavior for precedence supporting routers is already specified in [Bak95], the router requirements RFC. According to this document, routers must order their output queues based on highest precedence. If packets need to be dropped, lower-priority packets must be dropped first. Routers must also select appropriate service levels of the lower layers to provide preferential treatment. However, strict priority ordering is not fair, because lower-priority traffic may be starved. *Weighted round-robin scheduling* solves the fairness problem. However, round-robin mechanisms work on a per-packet basis and do not take the size of packets into account, thus they do not distribute bandwidth exactly according to the weights.

Weighted RED is a mechanism that combines characteristics of fair scheduling and RED. While the queue is not full, weighted RED does nothing. When the queue grows too long, weighted RED selects a packet to be dropped according to its precedence class. Weighted RED can be efficiently done on an off-board processor if the number of classes is fixed. Thus, weighted RED causes TCP sources to back of and slow-start and it ensures that UDP traffic of a certain precedence does not exceed its fair bandwidth share during periods of congestion.

The disadvantage of weighted RED is that the precedence of a packet only influences the dropping decision, not the scheduling decision under normal load. A high-precedence packet will still get queued behind low-precedence packets. To ensure that higher-precedence traffic also gets better service under normal load, weighted RED is combined with *class-based queuing* (CBQ). CBQ is a hierarchical version of weighted fair queuing (WFQ). If the number of classes is fixed, CBQ can be implemented efficiently. CBQ takes the previously used bandwidth into account to dynamically determine which packet to send next. High-precedence packets are treated preferentially as long as the bandwidth assigned to that precedence class is not exceeded. Over time, each traffic class gets a fair share of the bandwidth during congestion. In contrast to the requirements in [Bak95], CBQ and weighted RED do not not starve lower-precedence traffic.

[HR96] proposes a *path precedence discovery mechanism*. A sender uses this mechanism to find out if the requested priority level is supported on the entire path. The consumer can then decide whether the expected benefit is worth requesting and paying for priority service. The precedence discovery mechanism works as follows. The sender sends a packet with the desired precedence to the receiver. If any of the routers along the packet's path are configured to administratively disallow forwarding packets with that precedence, the router discards the packet and returns an ICMP Destination Unreachable message with a (new) code meaning "precedence not allowed". When a host receives such a destination unreachable message, it reacts according to the requirements of the application. For example, the host might either probe for a lower priority or decide not to communicate at that time.

Internet service providers (ISPs) need to agree on a common structure for precedence requests. Bi- and multilateral agreements ensure that precedence requests of customers of other ISPs are honored in the same way as precedence requests of the ISP's own customers. A consumer can use the path precedence discovery mechanism to find out if the path honors certain precedence levels. This kind of contract is much simpler than the contracts needed to support RSVP quality of service across several provider networks.

Precedence-based quality of service has several significant advantages. First, precedences are conceptually simple. There is no control protocol, and all decisions are made locally. The packet itself contains all the information that is necessary to schedule the packet. No additional state is needed in the routers. Precedence-based QoS fits well into the connectionless, stateless IP model. Second, precedence scheduling is relatively inexpensive to implement. The classification overhead for each packet is limited by the time it takes to read the precedence header field (e.g. the TOS field). Packet scheduling costs depend on the number of different precedence classes, but since this number is small and fixed, optimizations are possible. In contrast to packet scheduling in the Integrated Services architecture, packets do not have to be sorted according to their required QoS. They arrive "pre-sorted" since the precedence class clearly specifies the appropriate queue. There is no overhead for admission control. At the same time, the lack of admission control is the most severe shortcoming of precedence-based quality of service. There is no mechanism that would prevent a precedence class from becoming overloaded. Even the highest precedence class offers merely best-effort service within the class. If too many subscribers to the high-precedence service contend for the bandwidth, packets are dropped and delay as well as delay variation increases. Therefore, the end-to-end quality depends on other subscribers and cannot be predicted.

Despite the lack of predictability and flexibility, precedence-based quality of service has the potential to significantly improve the performance of real-time applications. It is widely viewed as the only strategy that is feasible today on a large scale (e.g. [Jac97]). The first step is the implementation of two service classes, regular and enhanced service. These service classes are associated with entries 0 and 1 in the TOS field. During this first step, QoS experience will be gained with a manageable system. Upgrading to up to eight precedence levels will later be little more than an administration issue.

6.4 Static resource allocation in advance

Resource sharing leads to better utilization of bandwidth and is thus cheaper than dedicated lines. The setback is that performance suffers during periods of congestion. Static resource allocation combines the advantages of resource sharing with the advantages of a dedicated line, at the loss of flexibility. *Virtual private networks* (VPNs) are set up by statically allocating resources on a path. The underlying technology is usually ATM or frame relay. A permanent virtual circuit (PVC) is configured between a sender and a receiver. If subscribers don't use their entire contracted bandwidth, the service provider can use the bandwidth for other traffic. Within their contracted bandwidth, subscribers get a service that resembles that of a dedicated line of that capacity. (A point of confusion arises because the term virtual private networks sometimes refers only to the privacy aspect of VPNs. Here, we refer only to the aspect of bandwidth guarantees.) There are certainly applications for virtual private networks. If access to the VPN is controlled within the subscriber, a VPN can ensure the quality of a video conference to a branch office across the country. However, virtual private networks are not a general solution to the quality of service problem because the PVC has to be set up at least hours in advance.

Class-based queuing (CBQ) is a mechanism that can be used to statically reserve a portion of a single link. CBQ guarantees a certain share of the bandwidth for each class during congestion but also provides sophisticated rules on how bandwidth can be borrowed between related classes. There is no setup protocol for CBQ. The bandwidth shares and hierarchical relationships are statically configured into routers with the help of a network management protocol. Thus, CBQ enables a company or institution to permanently reserve a bandwidth share on a certain link. This is useful in cases where the link has been identified as a bottleneck, and the overall quality can be significantly improved by eliminating congestion for one's own traffic at that one link.

Several router vendors offer customizable queuing options for their routers. One option are simple priorities, another option is CBQ-like queuing. So far, packets are commonly filtered and scheduled according to the protocol type. The customizing is done statically by the network administrator. More flexibility is needed to make this approach a true alternative to Integrated Services. Allocating resources on single links is an even less general approach than virtual private networks, but there is certainly a market for this strategy.

6.5 ST-II

ST-II (*Revised Internet Stream Protocol*) is an earlier resource reservation protocol for point-to-multipoint communication. A recent revision of ST-II is ST2+, specified in [DB95]. Both ST-II and RSVP were developed to support the ISPN service model. Both protocols set up reservations to provide end-to-end quality of service. Both support unicast as well as multicast. However, the protocols are inherently different. First, ST-II uses a connection-oriented protocol to set up reservations. Second, the sender is involved in every group action. In dynamically changing multicast groups, receivers wishing to be added to the group send a *join* message to the sender. The sender then explicitly adds the receiver to the multicast tree and to the reservation with an *add* message. Third, ST-II does not support heterogeneous reservations. All receivers get the same QoS. The common reservation is the least common denominator - all receivers get the QoS requested by the least capable or least demanding receiver. When a new receiver joins the group and requests an even smaller reservation, either the reservation for the entire group is scaled back or the new receiver gets a *drop* message. Fourth, ST-II achieves reliability and robustness using a complicated failure detection mechanism, while RSVP relies on soft state.

In summary, ST-II is a very complex protocol that provides fewer capabilities than RSVP. Because a ST-II sender needs to keep track of all receivers, ST-II does not scale with the number of receivers in a multicast group. Both the reservation setup and the failure detection protocols require a lot of control traffic. The other scaling problems of resource reservation, reservation state management and packet classification and scheduling, remain unsolved. Therefore, ST-II does not have significant advantages over RSVP.

Chapter 7

Conclusions

Integrating different traffic types in one robust and scalable packet-switched network is highly desirable. However, the challenge is that real-time applications have different service requirements than other applications. The traditional service model offers only best-effort service, which works very well for a large number of applications that don't have strict timing requirements. But best-effort service may be inappropriate for highly delay-sensitive ("real-time") traffic. Our observations on audio- and video transmissions (described in Sections 3.3 and 3.4) show that delay variation and packet loss severely diminish the quality of many MBone real-time transmissions. Because low perceived quality reduces the incentive to buy real-time applications, a solution to the quality problem has to be found before real-time applications for the Internet can become widely used in production environments. In addition to the true need for better service quality for real-time applications, service providers also feel the need to offer quality of service features in order to distinguish themselves from their competitors.

The evolving Integrated Services architecture implements a strategy to integrate all traffic types on one network and to differentiate between different services and service requirements. The Integrated Services architecture aims to provide predictable quality of service for real-time traffic. With Integrated Services, the quality of service which an application receives depends only on the type and size of its resource request, not on the current traffic situation. The resource reservation protocol RSVP is part of this architecture. A network can only guarantee some level of quality of service if it can protect itself from being overloaded. The network needs a way to commit to a certain amount of traffic and reject all other traffic. RSVP is a means to communicate control information about committed resources across the network and enables each router to do admission control. RSVP is designed to be general and flexible and can be extended to carry a wide variety of QoS requests.

It is not an objective of RSVP and Integrated Services to provide better quality of service for short-lived, non-realtime applications like ftp, telnet, WWW or email. Although interactive non-realtime applications also suffer from congested networks, their short lifetime does not justify the overhead of setting up and tearing down a reservation. The quality of service for these applications needs to be improved by providing adequate resources, or by deploying another form of differentiated service.

The RSVP working group positions RSVP as one important mechanism to provide differentiated quality of service that will coexist with other mechanisms on the future Internet. RSVP version 1 is a first step towards resource reservation on the global Internet. The working group does not recommend to deploy RSVP on a large-scale basis yet, because many building blocks of Integrated Services are still in the design phase. These include accounting, authorization and policy control modules that are necessary to prevent security attacks on open networks. However, the working group encourages institutions to deploy RSVP and Integrated Services on intranets, where scalability, security and access policies are not such critical issues. Intranet multimedia applications will likely be the first to benefit from RSVP.

In this work, we analyzed the applicability of RSVP. We found that

1. Guaranteed service has very limited applicability.
2. RSVP/IS in intranets is feasible today, but the benefits might not justify the costs.
3. RSVP on the global Internet requires aggregation.
4. Class of Service approaches are more feasible in the near future.

These results are explained in the following paragraphs.

1. Applicability of guaranteed service

In contrast to controlled load service, guaranteed service claims to offer quantitative delay guarantees. Our experiments on router-induced jitter and other sources of delay variation show that it is not feasible to guarantee a meaningful bound on delay. There are three reasons for this claim. First, we have shown that routers periodically delay packets under certain circumstances (Section 3.2.3). These circumstances are difficult to predict but lead to significant delays. The observed behavior is just one example for the unpredictability of latency in network elements. In spite of this unpredictability, guaranteed service requires that the worst-case delay in a network element is known and can be exported.

Second, the worst-case delay for the entire path is approximated by the sum of the worst-case delays of each network element. The probability of a packet being maximally delayed in every network element on the path is very small. However, in order to truly guarantee a bound on delay, the approximation must be conservative. The total worst-case delay reflects a coincidence of many exceptional situations, such as interference of routing updates or garbage collection with packet forwarding. The worst case must also be assumed for link layer scheduling delays because they cannot be controlled by guaranteed service. Thus, the total worst-case delay can easily add up to several seconds. A guaranteed bound on delay in the order of several seconds is completely useless for an application that chose guaranteed service because of strict real-time requirements.

A third reason why a bound on delay is not always meaningful is that this bound can only reflect network delay. Our experiments show that some senders of real-time traffic also cause significant delay variation (Section 3.2.2). Even if the network delay was within the guaranteed bound, the overall quality of service would still be unsatisfactory in these cases. To prevent host-induced delay variation, the host would need to run a real-time operating system.

Furthermore, it is generally impossible to aggregate guaranteed service flows. Scalability thus remains an unsolved issue.

Therefore, we believe that guaranteed service as specified in [SPG97] is not a promising building block of the architecture and will not be widely supported by vendors and Internet service providers.

2. RSVP in intranets

The deployment of RSVP and Integrated Services in intranets is viewed as the first useful step towards widespread use. On intranets, Integrated Services can protect multimedia conferences from disturbances caused by bursty data traffic. Even though company backbones usually don't suffer from long-lasting congestion, high average data rates plus occasional bursts can lead to high jitter and some packet loss. In Section 3.4, we showed that even flows with low packet loss rates can have high jitter. RSVP in conjunction with appropriate packet scheduling works well to reduce jitter significantly, though not completely. Our experiments on the effect of reservations on jitter showed that a reservation reduced delay variations to about a third when measured on a heavily used link with a bursty load. Reservations also completely eliminate packet loss, as measured in Section 4.2.

RSVP is a flexible way to protect real-time traffic on a small serial connection to an Internet provider. RSVP could be useful in such a situation even though the end-points of the serial line might be the only RSVP-capable routers on the path other than source and receiver.

Deployment of RSVP in intranets avoids many problems that arise in the global Internet. Within private networks, policy control does not play such a critical role. The security risk imposed by RSVP (such as the risk of denial-of-service attacks) is tolerable within private networks. Within an intranet, scaling is less important. Furthermore, intranets are usually contained within a single domain. There is no issue of competing providers and related compatibility problems.

On the other hand, unsatisfactory replay quality of real-time applications is more likely caused by a congested link in the global backbone than by events that occur within the intranet. Intranets usually have sufficient resources for normal demand. If they don't, the appropriate action is to add bandwidth and to upgrade routers in order to provide potential for growth. Thus, the main reason for deploying RSVP in intranets is to guarantee some level of quality of service during peak demand. However, in many cases it is impossible to give end-to-end guarantees because the LANs of sender and receiver don't support guarantees (as discussed in Section 4.3).

Thus, deployment of RSVP in intranets is feasible with today's technology and has the potential to improve the playback quality of real-time applications in some situations. However, the need for guarantees is not entirely compelling and the QoS for real-time applications within intranets could also be enhanced through simpler means, such as adding bandwidth or precedence-based differentiated service.

3. RSVP on the global Internet

The ultimate goal of RSVP is to provide real-time applications with a predictable quality of service across networks of every size and technology. This scenario requires that Integrated Services are supported by virtually every router. Under the assumptions that the real-time portion of Internet traffic will grow significantly in the next few years and users will have widespread access to RSVP, the number of reservations will skyrocket. At that point, the poor scaling properties of RSVP (see Section 4.5) will become an obstacle for any further growth of the network. Routers store information about every reservation and look up the reservation state for every packet they forward. The cost of packet forwarding grows faster than the bandwidth of the outgoing link because of the increased number of reservations and different services on a large link. Thus, it is impossible to deploy RSVP on a large scale as a protocol that reserves resources for single application-level flows. Hence, some form of flow aggregation is necessary. Aggregation means that several flows are treated as one. If it can be ensured that all individual flows behave well and sufficient resources are available for the aggregated flow, aggregation can help to provide predictable QoS in a scalable way. However, aggregate admission control is difficult to do without abandoning no-loss guarantees or wasting bandwidth, as discussed in Section 5.1. Furthermore, part of the flexibility of RSVP is lost. RSVP allows applications to specify their own QoS parameters. With aggregation, only a fixed number of different QoS levels can be supported and flows with slightly different QoS requirements are grouped together.

Layer 3 switching in connection with RSVP seems to raise as many new problems as it solves. We thus don't expect proprietary switching technologies such as tag switching and IP switching to become general solutions for RSVP's scaling problems.

In summary, we don't see a complete solution to RSVP's scaling problems that preserves all desirable properties of RSVP. If one is willing to give up 100% guarantees (which are likely to be impossible in the Internet in any case), aggregation is a valid strategy to make RSVP scalable.

4. RSVP/Integrated Services versus Class of Service approaches

Most researchers, content providers and users in the Internet community agree that some form of differentiated service is highly desirable in the Internet. However, it is controversial whether Integrated Services (IS) is the goal that should be pursued or if a simpler mechanism can achieve the same improvements in service quality.

Differentiated service is needed to support the inherently different service requirements of real-time applications and elastic applications. Class of service (CoS) mechanisms offer a small fixed number of service classes. Each packet carries an identifier specifying the requested service class. Packets are scheduled based on this identifier. In contrast, an IS packet scheduler filters packets according to various fields in the layer 3 and layer 4 packet headers. It looks up the specific QoS that a flow requested and then schedules the packet accordingly. Because there are an unknown number of different QoS requirements, packet scheduling requires sorting, which is costly. CoS mechanisms allow routers to concentrate on their primary function, namely packet forwarding. No valuable computing and memory resources are spent on flow state management and sorting. Thus, CoS routers will be less expensive than IS routers.

The fundamental difference between IS and CoS approaches lies in the guarantees they give. Controlled load service as specified by the IS working group guarantees zero packet loss caused by overload. Delay variations are guaranteed to closely resemble those in an uncongested network.

The guarantees given by CoS mechanisms are much weaker. CoS approaches guarantee that packets with higher precedence get better service than packets with lower precedence. There is no admission control, thus there is no mechanism to prevent classes from becoming overloaded. Within each class, packets get best-effort service. Therefore, the quality of service is not predictable but depends on the amount of competing traffic in that class. Congestion is possible within classes and leads to jitter and packet loss. However, single QoS classes are completely isolated. If bursty data traffic and real-time traffic can be successfully separated by precedence, bursty data traffic does not cause jitter for real-time applications and data traffic is not starved by an excessive amount of real-time traffic.

The key to success with CoS packet scheduling is sufficient bandwidth for high-precedence classes. Congestion within precedence classes that are chosen by real-time applications must be avoided. If enough bandwidth is allocated to the higher-precedence classes so that real-time traffic does not experience congestion, adaptive playback applications will be able to make the remaining network jitter unnoticeable.

Outlook

Since the early work on RSVP in 1993 ([ZDE$^+$93]), researchers have gained much experience with quality of service and Integrated Services (IS). Many problems have been solved and other new problems have emerged. Integrated Services and RSVP is a very general approach to guarantee quality of service for real-time applications. IS/RSVP can guarantee a certain bandwidth for a flow and can keep network jitter that is caused by queuing within certain bounds. However, IS/RSVP doesn't have complete control over jitter. Quantitative guarantees are almost impossible to give because of the variety of hardware technology and networking protocols which comprise the Internet. Without quantitative guarantees, RSVP does not have much to offer that cannot be provided by precedence-based differentiated service. We expect that QoS development in the near future will shift to the much simpler Class of Service mechanisms. If experience shows that precedence-based services cannot provide the QoS customers want, some form of resource reservation might play a role in a future more mature Internet.

Appendix A

List of Acronyms

ATM	Asynchronous Transfer Mode
CBQ	Class-Based Queuing
CIDR	Classless Inter-Domain Routing
CoS	Class of Service
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DNS	Domain Name Service
FIFO	First in First out
FTP	File Transfer Protocol
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IS	Integrated Services
ISP	Internet Service Provider
ISPN	Integrated Services Packet Network
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
MAC	Media Access Control
MBone	Multicast Backbone
NFS	Network File Service
PCM	Pulse Code Modulation
PIM	Protocol-Independent Multicast
PVC	Permanent Virtual Circuit

QoS	Quality of Service
RED	Random Early Detection
RFC	Request for Comments
RSVP	Resource ReSerVation Protocol
RTCP	Real-Time Control Protocol
RTP	Real-time Transport Protocol
SMDS	Switched Multimegabit Data Service
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
ST-II	Revised Internet Stream Protocol
SVC	Switched Virtual circuit
TCP	Transmission Control Protocol
TOS	Type of Service
UDP	User Datagram Protocol
VC	Virtual Circuit or Channel
VPN	Virtual Private Network
WAN	Wide Area Network
WFQ	Weighted Fair Queuing

Bibliography

[Ada] R. Brian Adamson. MGEN-1.8b: NRL Unicast/Multicast Network Performance Measurement Tool Set. (adamson@itd.nrl.navy.mil). NRL Code 5523/ NEWLINK Global Engineering Corporation.

[Bak95] F. Baker. RFC 1812: Requirements for IP Version 4 routers, June 1995.

[BB97] S. Berson and L. Berger. IP Integrated Services with RSVP over ATM. Internet Draft, Mar 1997. Work in progress. draft-ietf-issll-atm-support-03.txt.

[BCS94] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. Request for Comments (Informational) RFC 1633, Internet Engineering Task Force, June 1994.

[BKS97] F. Baker, J. Krawczyk, and A. Sastry. RSVP Management Information Base. Internet Draft, June 1997. Work in progress. draft-ietf-rsvp-mib-08.txt.

[Boy97] Jim Boyle. RSVP Extensions for CIDR Aggregated Data Flows. Internet Draft, June 1997. Work in progress. draft-ietf-rsvp-cidr-ext-01.txt.

[BV97] S. Berson and S. Vincent. A "Classy" Approach to Aggregation for Integrated Services. Internet Draft, Mar 1997. Work in progress. draft-berson-classy-approach-00.txt.

[BZ96] R. Braden and L. Zhang. Resource ReSerVation Protocol (RSVP) – Version 1 Message Processing Rules. Internet Draft, Oct 1996. Work in progress. draft-ietf-rsvp-procrules-00.txt.

[BZ97] B. Braden and L. Zhang. Resource ReSerVation Protocol (RSVP) – Version 1 – Analysis and Applicability. Unpublished Internet Draft, Mar 1997.

[BZB$^+$97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. Internet Draft, Jun 1997. Work in progress. draft-ietf-rsvp-spec-16.txt.

[CDF$^+$95] D. Clark, S. Deering, D. Ferrari, C. Huitema, and S. Shenker. Reservations or no Reservations, Apr 1995. Infocom '95 Panel (Slides), ftp://ftp.parc.xerox.com/pub/net-research/infocom95.html.

[CSZ92] David D. Clark, Scott Shenker, and Lixia Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network; Architecture and Mechanism. *ACM SIGCOMM '92*, 22(4):14–26, August 1992.

[DB95] L. Delgrossi and L. Berger. RFC 1819: Internet Stream Protocol Version 2 (ST2) Protocol Specification — Version ST2+, August 1995.

[Dee91] S. E. Deering. *Multicast Routing in a Datagram Internetwork.* Ph.D Thesis, Stanford University, 1991.

[DEF$^+$97] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, A. Helmy, and L. Wei. Protocol Independent Multicast Version 2, Dense Mode Specification. Internet Draft, May 1997. Work in progress. draft-ietf-idmr-pim-dm-05.txt.

[Dow96] Kevin Dowd. *Getting connected to the Internet at 56k and up.* Oreilly & Assoc, 1996.

[FJ95] Sally Floyd and Van Jacobson. Link-sharing and Resource Management Models for Packet Networks. *IEEE/ACM Transactions on Networking, Vol. 3 No.4, pp. 365-386*, August 1995.

[GPS97] A. Ghanwani, J. W. Pace, and V. Srinivasan. A Framework for Providing Integrated Services Over Shared and Switched LAN Technologies. Internet Draft, April 1997. Work in progress. draft-ietf-issll-is802-framework-01.txt.

[Her97] Shai Herzog. RSVP Extensions for Policy Control. Internet Draft, March 1997. Work in progress. draft-ietf-rsvp-policy-ext-02.txt.

[HR96] G. Huston and M. Rose. Path Precedence Discovery. Internet Draft, December 1996. Work in progress. draft-huston-pprec-discov-00.txt.

[Hui95] Christian Huitema. *Routing in the Internet.* P T R Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1995.

[IP 96] IP Switching: The Intelligence of Routing, the Performance of Switching, February 1996. Ipsilon Networks white paper. http://www.ipsilon.com/productinfo/wp-ipswitch.html.

[Jac88] Van Jacobson. Congestion Avoidance and Control. In *Proceedings, SIGCOMM '88 Workshop*, pages 314–329. ACM SIGCOMM, ACM Press, August 1988. Stanford, CA.

[Jac94] Van Jacobson. Multimedia conferencing on the Internet, August 1994. SIGCOMM '94 conference tutorial.

[Jac95] Van Jacobson. The MBone - Interactive Multimedia on the Internet, Feb 17, 1995. Slides.

[Jac97] Van Jacobson. Quality of Service for the Next Generation Internet. White paper and talk at the Workshop on Research Directions for the Next Generation Internet, May 13-14, 1997, Vienna, VA, USA. Computing Research Association (CRA), May 1997. http://www.cra.org/Policy/NGI/papers/jacobsonWP.

[JDSZ95] Sugih Jamin, Peter Danzig, Scott Shenker, and Lixia Zhang. A Measurement-based Admission Control Algorithm for Integrated Services Packet Networks. *ACM SIGCOMM 95*, Aug 1995.

[JM] Van Jacobson and Steve McCanne. Visual audio tool (vat). Software online, ftp://ftp.ee.lbl.gov/conferencing/vat.

[Kum95] V. Kumar. *MBone, Interactive Multimedia on the Internet.* New Riders Publishing, Indianapolis, Indiana, 1995.

[KWTL97] J. Krawczyk, J. Wroclawski, A. Terzis, and L.Zhang. RSVP tunneling support, April 1997. Slides. IETF Memphis. ftp://ftp.isi.edu/rsvp/memphis.ietf/zhang-tunnel.slides.ps.

[MBB$^+$97] A. Mankin, F. Baker, B. Braden, S. Bradner, M. o'Dell, A. Romanow, A. Weinrib, and L. Zhang. Resource ReSerVation Protocol (RSVP) – Version 1 Applicability Statement: Some Guidelines on Deployment. Internet Draft, Mar 1997. Work in progress. draft-ietf-rsvp-intsrv-analysis-00.txt, see also [BZ97].

[Mit95] D. J. Mitzel. *A Study of Resource Reservation Protocol Scaling and Dynamics in Integrated Service Packet Networks.* Ph.D Thesis, University of Southern California, Dec 1995.

[MJ95] Steve McCanne and Van Jacobson. vic: A Flexible Framework for Packet Video. *ACM Multimedia '95*, 1995.

[MS97] T. Maufer and C. Semeria. Introduction to IP Multicast Routing. Internet Draft, March 1997. Work in progress. draft-ietf-mboned-intro-multicast-01.txt.

[NEH$^+$96] P. Newman, W. L. Edwards, R. E. Hoffman, F. Ching Liaw, T. Lyon, and G. Minshall. RFC 1953: Ipsilon Flow Management Protocol Specification for IPv4 Version 1.0, May 1996.

[Net96] NetFlowSwitching and Services: Extending Today's Internetwork to Meet Tomorrow's Requirements, April 1996. Cisco Systems product overview. http://www.cisco.com/warp/public/733/packet/nfss_ ov.htm.

[Nol96] Thomas Nolle. Reservations about RSVP. *NetworkWorld*, October 28, 1996.

[PG93] A. K Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case. In Michael G. Hluchyj, editor, *Proceedings of the 12th Annual Joint Conference of the IEEE Computer and Communications Societies on Networking: Foundations for the Future. Volume 2*, pages 521–530, Los Alamitos, CA, USA, March 1993. IEEE Computer Society Press.

[Pos81] J. Postel. RFC 793: Transmission Control Protocol, September 1981.

[Ram96] Sanjeev Rampal. Flow Grouping For Reducing Reservation Requirements for Guaranteed Delay Service. Internet Draft, December 1996. Work in progress. draft-rampal-flow-delay-service-00.txt.

[RDK+97] Yakov Rekhter, Bruce Davie, Dave Katz, Eric Rosen, George Swallow, and Dino Farinacci. Tag Switching Architecture - Overview. Internet Draft, Jan 1997. Work in progress. draft-rekhter-tagswitch-arch-00.txt.

[SCFJ96] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889: RTP: A Transport Protocol for Real-Time Applications, January 1996.

[Sch96] H. Schulzrinne. RFC 1890: RTP Profile for Audio and Video Conferences with Minimal Control, January 1996.

[SPG97] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. Internet Draft, July 1997. Work in progress. draft-ietf-intserv-guaranteed-svc-08.txt.

[Sta88] William Stallings. *Data and computer communications (2nd ed.)*. Macmillan, New York, 1988.

[Wro97] J. Wroclawski. Specification of the Controlled-Load Network Element Service. Internet Draft, May 1997. Work in progress. draft-ietf-intserv-ctrl-load-svc-05.txt.

[ZDE+93] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Networks*, September 1993.